ORC 67-6
MARCH 1967

COMPLEX: A COMPLEMENTARY SLACKNESS, OUT-OF-KILTER ALGORITHM FOR LINEAR PROGRAMMING

by

William S. Jewell

OPERATIONS RESEARCH CENTER

COLLEGE OF ENGINEERING

UNIVERSITY OF CALIFORNIA - BERKELEY

96

COMPLEX:
A COMPLEMENTARY SLACKNESS, OUT-OF-KILTER ALGORITHM FOR LINEAR PROGRAMMING

by

William S. Jewell
Department of Industrial Engineering
and Operations Research
University of California, Berkeley

March 1967 ORC 67-6

This research has been partially supported by the Office of Naval Research under Contract Nonr-222(83), the National Science Foundation under Grant GP-7417, and the U.S. Army Research Office-Durham, Contract DA-31-124-ARO-D-331 with the University of California. Reproduction in whole or in part is permitted for any purpose of the United States Government.

ABSTRACT

An algorithm has been developed which uses the complementary slackness principle to take completely arbitrary primal and dual solutions to a linear program with doubly-bounded variables into the optimal solutions. The algorithm is not a new method; viewed in the proper context, it can be thought of either as an elaboration of the primal-dual, composite, breakpoint-tracing, or complementary pivot algorithms: as an extension of the out-of-kilter or black-box methods for network flows; or, finally, even as a special way of looking at the original simplex algorithm. However, it possesses certain pedagogical advantages:

1. Proper emphasis is placed on complementary slackness as the fundamental constructive principle of linear programming, and infeasibility and unboundedness are related to secondary roles.

2. Arbitrary starting solutions are allowed, and arbitrary lower and upper bounds on the variables are handled naturally.

3. One activity at a time is "worked on;" complementary slackness always indicates what operations are necessary; no artificial distinction is made between "real," "artificial," or "slack" variables.

4. The imbedded linear program is of an extremely simple type, which reveals the essential nature of simplifications which can be made in models of special structure.

5. Very few set-theoretic proofs and tableaux rules are needed, almost all operations being described on the optimality diagram for each activity.

Almost all of the simpler procedures, such as Phase I, the dual simplex method, parametric programming, the primal-dual algorithm, etc. can be viewed as special cases of the complex algorithm which use special starting solutions and special heuristics.

TABLE OF CONTENTS

COMPLEX:

A COMPLEMENTARY SLACKNESS, OUT-OF-KILTER

ALGORITHM FOR LINEAR PROGRAMMING

by

William S. Jewell

0. INTRODUCTION

The purpose of this paper is to present a *complementary slackness, out-of-kilter algorithm* for the following dual linear programs:

$$\text{Minimize } C = \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad (i = 1,2, \ldots, m) \tag{0.1}$$

$$\ell_j \le x_j \le k_j \qquad (j = 1,2, \ldots, n)$$

$$\text{Maximize } B = \sum_{i=1}^{m} b_i y_i - \sum_{j=1}^{n} k_j u_j + \sum_{j=1}^{n} \ell_j w_j$$

$$y_i \text{ unrestricted}$$

$$u_j, w_j \ge 0 \tag{0.2}$$

$$\sum_{i=1}^{m} y_i a_{ij} - u_j + w_j = c_j$$

where no restriction is made on the constants except $\ell_j \le k_j$ for all j. (For a symmetric form, see Section 9.)

The algorithm to be presented is not a new method; viewed in the proper light it can be thought of either as an elaboration of the primal-dual, composite, breakpoint tracing, or complementary pivot algorithms; as an extension of the out-of-

kilter or black-box methods for network flows; or finally, even as a special way of looking at the original simplex algorithm. In fact, each of these methods can be produced by the use of appropriate heuristics within the procedure presented in Section 6.

Why, then, another algorithm? When teaching linear programming, one is struck by the fact that almost all "advanced" topics are merely repetitions of the same basic concepts of *pivoting* (to move from one extreme point to another) and *pricing-out* (to determine a desirable direction in which to move), with slightly different emphasis on which variable is to be increased or decreased, and by how much that and other variables are allowed to change. What is needed, it seems, is a general algorithmic framework in which all the various extreme-point procedures can be explained as variants of a common procedure which use special heuristics.

In the author's opinion, such a *common simplex* procedure should emphasize the following points:

(0.3)

(a) Much greater emphasis should be placed on the *complementary-slackness* relationship as the *fundamental working principle* of programming; questions of feasibility and boundedness should be relegated to a secondary role;

(b) The simplex procedure is a *local*, or *incremental* move in which one basis change is made to improve some functional; the problem is to know how to make this one move and interpret the results--the remaining steps will always "look" identical;

(c) Arbitrary starting solutions should be allowed, and tedious conventions on signs of variables and constants, form of constraints, "real" versus "artificial" variables, etc. should be eliminated as much as possible;

(d) Less emphasis needs to be given to tableaux and a variety of formal rules for row and column manipulations; the algorithm and the current "state" of the solution should remind one of the correct rule to be used.

Given that the elements for such a development have been available since 1959 [7,8,11], it is remarkable how many variants and elaborations have been presented (often with intriguing prior arguments for efficiency), but how little unifying and

synthesizing work has been done. We do not pretend that the algorithm presented here will be a definitive one, but only hope that it will begin to reveal the underlying coherence and harmony between the various approaches, and how simple and straightforward a general theory is.

The report is in the following main groupings: the first four sections examine the central ideas of the optimality diagram and the incremental linear program subroutine; Sections 5-7 present the main algorithm and its proofs; Section 8 presents some of the many options available in using the algorithm, followed by a discussion of symmetric formulations in Section 9; Section 10 discusses the important special case of network flow problems; and Sections 11-13 conclude with a review of the basic ideas of the report, an appreciation of J. B. Dennis' important work, and a survey of the extensions which are possible. The Appendices present certain fine points, such as the organization of tableaux, and the interpretation of classical algorithms as special cases of the common procedure.

1. THE OPTIMALITY PRINCIPLE AND DIAGRAM

To fulfill point (0.3a), we begin by defining the constructive principle which will be used to solve (0.1) and (0.2) and express this principle as a set of n diagrams in $\mathbb{R}^2$, one for each activity.

Until Section 9, we assume:

(1.1) *every set of values of the primal variables* $x_j = x_j^o$ $(j = 1,2, \ldots, n)$ *will always satisfy the equality constraints* $\sum a_{ij}x_j^o = b_i$ $(i = 1,2, \ldots, m)$

either by adjoining *slack variables* of any sign in the formulation, or by adjoining *error variables* of any sign after the initial solution is chosen. These particular constraints will never be violated during the algorithm and can henceforth be ignored; however, x_j^o may exceed either of its bounds k_j or ℓ_j.

Let

$$z_j = \sum_{i=1}^{m} y_i a_{ij} \qquad (j = 1,2, \ldots, n) \tag{1.2}$$

be the *profitability* of activity j. For given initial values $\{x_j^o; y_i^o\}$ of the primal and dual variables satisfying (1.1), we define the *state of activity* j as the pair of values $(x_j^o; z_j^o)$, and partition the *state space* as shown in (1.3).

(1.3)

	$x_j^o < \ell_j$	$x_j^o = \ell_j$	$\ell_j < x_j^o < k_j$	$x_j^o = k_j$	$x_j^o > k_j$
$z_j^o > c_j$	$j \in K^-$			$j \in K$	$j \in K^+$
$z_j^o = c_j$	$j \in B^-$	$j \in B$			$j \in B^+$
$z_j^o < c_j$	$j \in L^-$	$j \in L$	$j \in L^+$		

Partition of State Space for Initial State $\{x_j^o; z_j^o\}$, of Activity j

For convenience, let

(1.4) $U^- = L^-$ or B^- or K^- ; $U = L$ or B or K ; $U^+ = L^+$ or B^+ or K^+ .

The state $\{x_j^t; z_j^t\}$ of a current solution at iteration t ($t = 0,1,2, \ldots$) can be shown most clearly on an *optimality diagram* $(x_j; z_j)$ for each activity j, shown in Figure 1.1; for example, U consists of the solid horizontal and vertical lines. Some of the states may be missing in degenerate cases.

The following is just a restatement of what is usually called the weak theorem of complementary slackness:

Optimality Principle

(1.5) *For a set of values* $\{x_j; z_j\}$, *satisfying* $\sum a_{ij}x_j = b_i$ ($i = 1, \ldots, m$) *to be optimal, it is necessary and sufficient that* $j \in U$ *for all activities* $j = 1, \ldots, n$.

(The strong theorem of complementary slackness states that there is at least one optimal solution which does not have any activities on the *corner points*, (B and $x_j = \ell_j$) , (B and $x_j = k_j$) ; however, we shall not need this fact in the sequel.)

An activity in U will be called *conforming* [14], or *in-kilter* [9]; a "non-U" activity in U^- or U^+ is *nonconforming*, or *out-of-kilter*.

The basic idea of the algorithm to be presented in Section 6 is as follows: The initial solution determines the state of all activities. An arbitrary non-conforming activity is selected, and changes in the variables are made by a sub-routine to make this selected activity more conforming (in a sense to be made precise in Section 7); these changes leave all currently conforming activities in-kilter, and no unselected nonconforming activity becomes more nonconforming.

The primary pedagogical advantage of the optimality diagram is that all features of the algorithm can be explained directly on the diagram; this simplifies notation

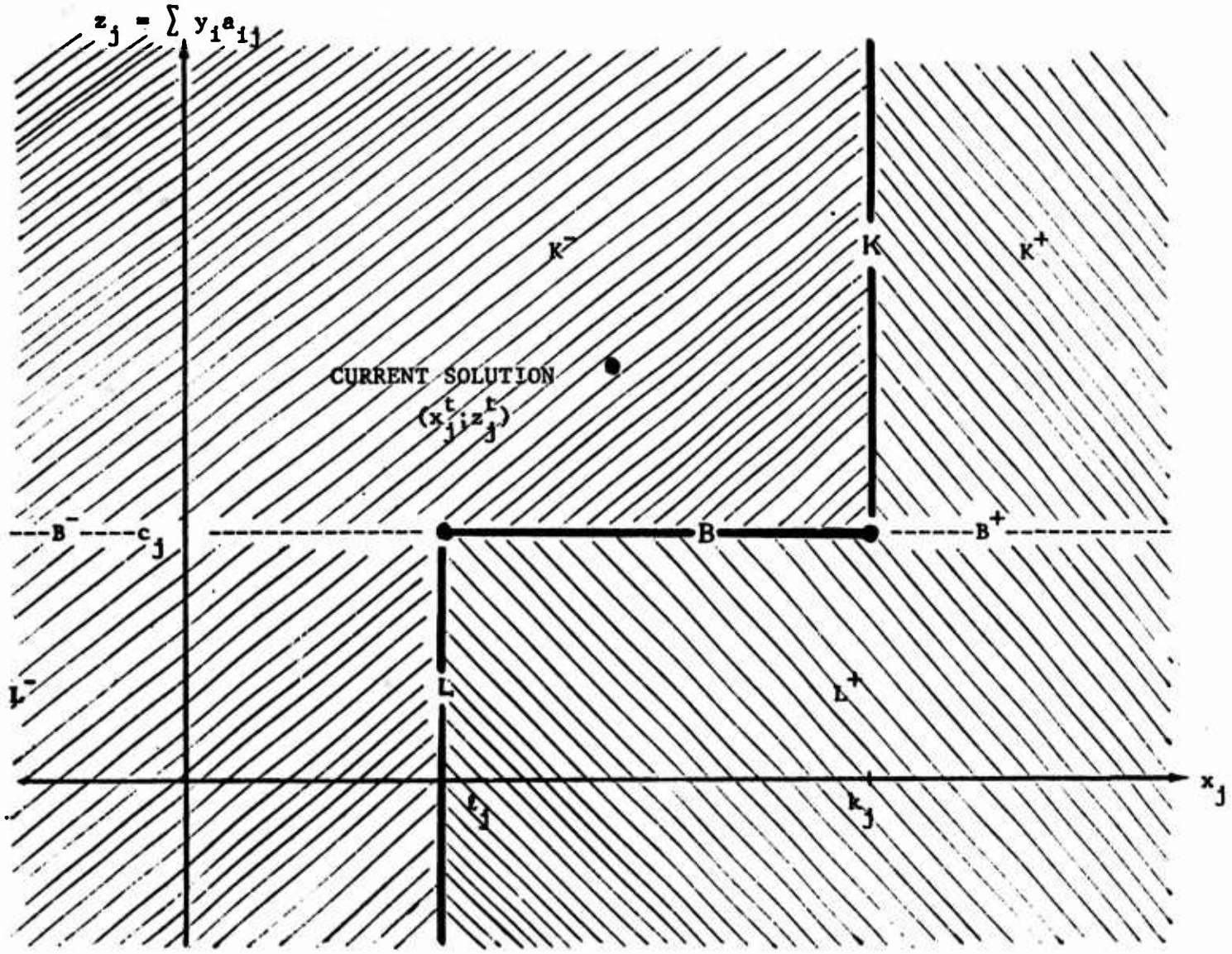


FIGURE 1.1: OPTIMALITY DIAGRAM FOR ACTIVITY j .

and provides a ready visualization and reminder of the rules of the algorithm, as well as suggesting various heuristic procedures.

2. THE INCREMENTAL LINEAR PROGRAM

The key procedure in the algorithm is a subroutine which determines the *incremental* changes to be made from the current solution. If $\{x_j^t\}$ are the values determined during the t^{th} $(t = 0,1,2, \ldots)$ iteration, the *incremental displacement*, ξ_j of activity j during the next iteration is

$$\xi_j = x_j - x_j^t \qquad j = 1, \ldots, n \tag{2.1}$$

The subroutine consists of the following *incremental linear program:*

$$\text{Maximize} \quad \Delta = \pm\xi_s$$

$$\sum_{j=1}^{n} a_{ij}\xi_j = 0 \qquad (i = 1,2, \ldots, m) \tag{2.2}$$

$$\lambda_j \leq \xi_j \leq \kappa_j \qquad (j = 1,2, \ldots, n)$$

$$\text{Minimize} \quad \sum_{j=1}^{n} (\kappa_j v_j - \lambda_j \omega_j)$$

$$\sum_{i=1}^{m} \eta_i a_{ij} - v_j + \omega_j = \begin{cases} 0 & j \neq s \\ \pm 1 & j = s \end{cases} \tag{2.3}$$

$$v_j \geq 0 \ , \ \omega_j \geq 0$$

$$\eta_i \quad \text{unrestricted}$$

The *selected activity index*, s, will be furnished to the subroutine, as will an independent set of column vectors, $\mathcal{J}$, from the matrix (a_{ij}). The values of the bounds provided will always be such that

$$-\infty \leq \lambda_j \leq 0 \leq \kappa_j \leq +\infty \ . \qquad (j = 1,2, \ldots, n) \tag{2.4}$$

So that the incremental origin is always (basic) feasible.

For the selected activity:

(2.5) if the problem is to *maximize* ξ_s , $\lambda_s = 0$; $0 < \kappa_s \leq +\infty$;
if the problem is to *minimize* ξ_s , $-\infty \leq \lambda_s < 0$; $\kappa_s = 0$.

The optimality diagram for the incremental program,

(2.6) $$\zeta_j = \sum \eta_i a_{ij} \quad \text{versus} \quad \xi_j ,$$

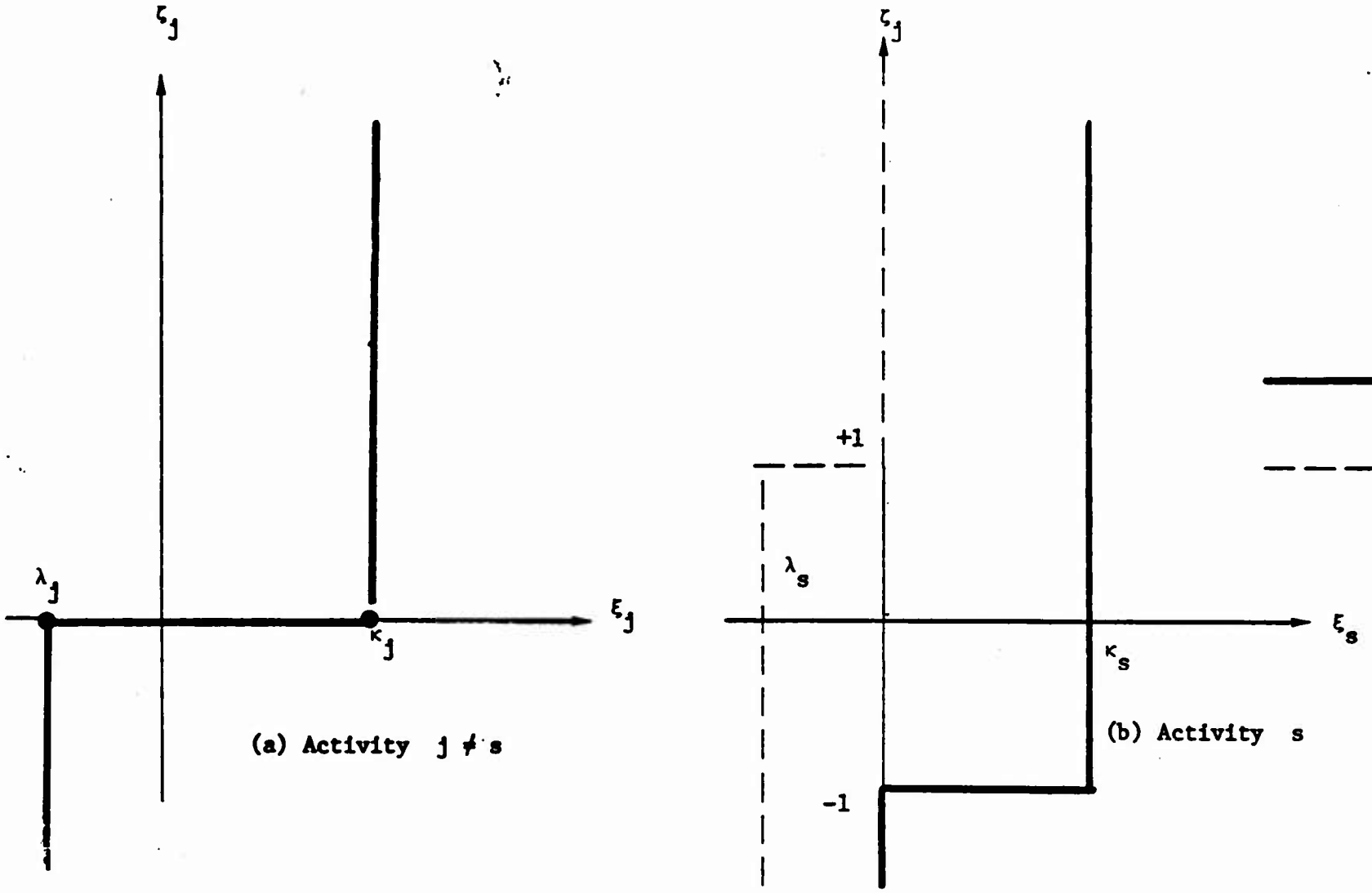
for activity j is shown in Figure 2.1. Note that some activities may have part or all of the ζ_j-axis or the ξ_j-axis as conforming states.

It follows from (2.4) and (2.5) that:

(2.7) *The values* $\xi_j = 0$, $j \in \mathcal{J}$, *constitute a basic feasible solution to (2.2) with* $\eta_i = 0$, $(i = 1,2, \ldots, m)$ *and* $\zeta_j = 0$, $(j = 1,2, \ldots, n)$ *as corresponding solutions to (2). (As usual,* $\xi_j = 0$, $j \notin \mathcal{J}$*). Furthermore, this basis and these values cannot be optimal solutions to (2.2) and (2.3).*

The initial incremental program is, in fact, in *complementary pivot form* (Section 9B). There are four possibilities for the optimal solutions, $\{\xi_j^*\}$, $\{\eta_i^*\}$, $\{\zeta_j^*\}$:

(2.8)
(a) The optimal value of ξ_s is unbounded $(\xi_s^* = +\infty$ or $-\infty)$.
(b) The optimal value of ξ_s reaches its nonzero bound $(\xi_s^* = \kappa_s$ or $\lambda_s)$.
(c) The optimal value of ξ_s is nonzero, and the displacement of some other activity reaches a bound $(0 < \xi_s^* < \kappa_s$ or $\lambda_s < \xi_s^* < 0$, and $\xi_j^* = \lambda_j$ or $\xi_j^* = \kappa_j$ for some $j \neq s)$.
(d) The optimal value of ξ_s is zero, but the given $\mathcal{J}$ is not the index set of the optimal basis.



(a) Activity $j \neq s$

(b) Activity s

FIGURE 2.1 (a) and (b): OPTIMALITY DIAGRAMS FOR THE INCREMENTAL PROGRAM

Appendix A discusses ways in which this special linear program can be organized in tableau form; Section 8 discusses simplifications which can be used for problems of special structure, and the possibility of selecting several activities.

In all cases, except (2.8a), we know from the optimality diagrams in Figure 2.1:

The finite optimal solutions to (2.2) and (2.3) satisfy:

$$\text{If } \lambda_j < \xi_j^* < \kappa_j \text{ , then } \zeta_j^* = \sum \eta_i^* a_{ij} = \begin{cases} 0 & j \neq s \\ \bar{+}1 & j = s \end{cases} ;$$

$$\text{if } \zeta_j^* \begin{matrix} (>) \\ (<) \end{matrix} \begin{cases} 0 & j \neq s \\ \bar{+}1 & j = s \end{cases} \text{, then } \xi_j^* = \begin{matrix} (\kappa_j) \\ (\lambda_j) \end{matrix} . \tag{2.9}$$

Obvious modifications apply when $\lambda_j = \kappa_j = 0$, or one or both of these is infinite.

In particular, we note that *not both* ξ_s^* and ζ_s^* can be simultaneously zero. In addition to providing the optimal displacement of the primal variables, we shall see in Section 4 that the incremental subroutine also furnishes the optimal *gradient* of dual displacement.

3. ACTIVITY SELECTION FOR THE INCREMENTAL LINEAR PROGRAM

The selected activity, s, which is to be "worked upon" in the incremental subroutine can be chosen from among *any* of the nonconforming activities which violate the Optimality Principle (1.5). Various heuristic procedures for this selection are discussed in Section 8.

For "primal-infeasible" activities, it is clear that the rule

$$\begin{aligned} &s \in L^-, B^-, \text{ or } K^- \text{ -- maximize } \xi_s ; \\ &s \in L^+, B^+, \text{ or } K^+ \text{ -- minimize } \xi_s , \end{aligned} \tag{3.1}$$

will move the state of activity s chosen towards feasibility. However, this rule is also correct if $\ell_s \leq x_s \leq k_s$ and $s \notin B$, as we shall see in the next section.

How far should the $\{\xi_j\}$ be allowed to move? Clearly, a conforming activity should not be allowed to leave U. Or, conversely, if some $j \notin U$, the incremental movement should be stopped when the activity reaches U. Finally, for finiteness, it is desirable to prevent "non-U" activities from becoming more so by moving counter to the rule (3.1).

This leads to the following rules for specifying the bounds $\{\lambda_j, \kappa_j\}$ for the $(t+1)^{st}$ application of the subroutine, in terms of the current values $\{x_j^t\}$ $(t = 0,1,2, \ldots)$.

(3.2) $(j = 1, \ldots, n)$

IF:	SET: λ_j	κ_j
$j \in L^-$	0	$\ell_j - x_j^t$
$j \in B^-, K^-$	0	$k_j - x_j^t$
$j \in L, K$	0	0
$j \in B$	$-(x_j^t - \ell_j)$	$k_j - x_j^t$
$j \in L^+, B^+$	$-(x_j^t - \ell_j)$	0
$j \in K^+$	$-(x_j^t - k_j)$	0

The directions of change and the allowed maximal increments are conveniently summarized on the Optimality Diagram in Figure 3.1. There are no infinite displacements allowed in the diagram as shown, but if $k_j = \infty$, or $\ell_j = -\infty$ for some j , it would be possible to obtain the unbounded solution of (2.8a).

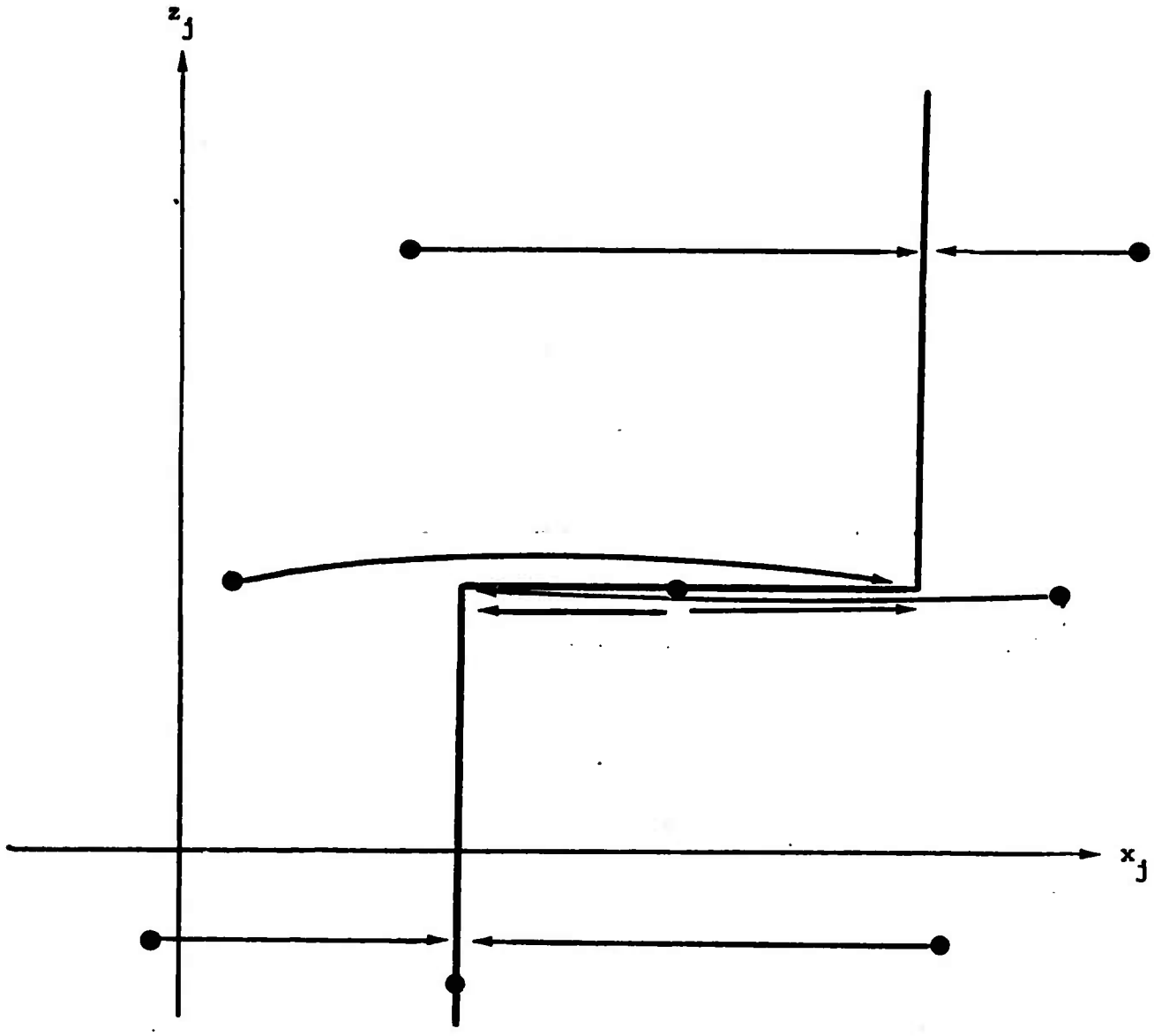


FIGURE 3.1: OPTIMALITY DIAGRAM FOR ACTIVITY j , SHOWING DIRECTION AND MAXIMAL PRIMAL DISPLACEMENTS ALLOWED FROM DIFFERENT STATES (●)

4. BREAKPOINT STEPPING WITH THE INCREMENTAL LINEAR PROGRAM

From the above discussion, the new activity levels $x_j^{t+1} = x_j^t + \xi_j^*$ obtained by adding the optimal incremental values from the subroutine to the old activity levels obviously keep all conforming activities so, and do not move nonconforming points farther away from or through the optimality diagram (see Section 7 for a discussion of conformity measures). Furthermore, the primal displacement only permits the following *limiting* changes of state from iteration t to $t+1$:

$$\text{(4.1)} \qquad \begin{array}{lll} \text{(a)}\ L^- \to L & \text{(d)}\ L^+ \to L & \\ \text{(b)}\ B^- \to B & \text{(e)}\ B^+ \to B & \left(\begin{array}{c} j \notin U \\ \text{at iteration } t \end{array}\right) \\ \text{(c)}\ K^- \to K & \text{(f)}\ K^+ \to K & \end{array}$$

or

$$\begin{array}{ll} \text{(g)}\ \left(x_j^t < k_j\right) \to \left(x_j^{t+1} = k_j\right) & \\ \text{(h)}\ \left(x_j^t > \ell_j\right) \to \left(x_j^{t+1} = \ell_j\right) & \left(\begin{array}{c} j \in B \\ \text{at iteration } t \end{array}\right) \end{array}$$

At least one such change will occur, possibly at a zero-change level.

Hopefully, the selected activity s might undergo a change of type (a) - (f), but in general, ξ_s may only move partway toward a conforming state, as shown by the horizontal line in Figure 4.2. In fact, from (2.8d) it is possible that $\xi_s^* = 0$.

After these horizontal (primal) movements are made on the various diagrams, *the dual solution to the incremental linear program furnishes the appropriate gradient for vertical (dual) changes through the formulae:*

$$\text{(4.2)} \qquad y_i^{t+1} = y_i^t + \theta\eta_i^* \qquad (i = 1,2, \ldots, m)$$

$$\text{(4.3)} \qquad z_j^{t+1} = z_j^t + \theta\zeta_j^* \qquad (j = 1,2, \ldots, n)$$

and appropriate selection of the *gradient step size,* θ .

Consider Figure 4.1. If the same requirements are placed on dual displacements as for the primal ones (all conforming activities remain so, and displacement stops when a nonconforming activity becomes conforming), then it follows from (2.9) and Figure 2.1 that *after the primal change* (4.3) we have:

(4.4)

IF:	THEN: ζ_j^*	θ
(a) $j\in L^-, B^-$, or (K^- and $x_j^{t+1}<\ell_j$)	$\zeta_j^*\leq 0$	unlimited
(b) $j\in(K^-$ and $\ell_j\leq x_j^{t+1}<k_j)$ or K	If $\zeta_j^*<0$	θ must be $\leq \dfrac{z_j^t-c_j}{-\zeta_j^*}$
(c) $j\in(L^+$ and $x_j^{t+1}>k_j), B^+$, or K^+	$\zeta_j^*\geq 0$	unlimited
(d) $j\in(L^+$ and $\ell_j<x_j^{t+1}\leq k_j)$ or L	If $\zeta_j^*>0$	θ must be $\leq \dfrac{c_j-z_j^t}{\zeta_j^*}$
(e) $j\in B$ and $\ell_j<x_j^{t+1}<k_j$	$\zeta_j^*=0$	unlimited
(f) $j\in B$ and $\ell_j=x_j^{t+1}$	$\zeta_j^*\leq 0$	unlimited
(g) $j\in B$ and $x_j^{t+1}=k_j$	$\zeta_j^*\geq 0$	unlimited

for every activity j. Changes which do not limit θ are shown as dotted lines in Figure 4.1. (In Section 7 it is shown that these displacements are also reducing nonconformity in a certain sense.)

We conclude that *all* conditions (4.4) are satisfied for all activities for every value of step size:

$$(4.5)\quad 0<\theta\leq\theta^*=\min\left\{\min_{j\ni(\ell_j\leq x_j^{t+1}\leq k_j)\ (\text{and } \zeta_j^*<0)}\left(\frac{z_j^t-c_j}{-\zeta_j^*}\right);\ \min_{j\ni(\ell_j\leq x_j^{t+1}\leq k_j)\ (\text{and } \zeta_j^*>0)}\left(\frac{c_j-z_j^t}{+\zeta_j^*}\right)\right\}$$

By selecting $\theta=\theta^*$, we guarantee that at least one change of state

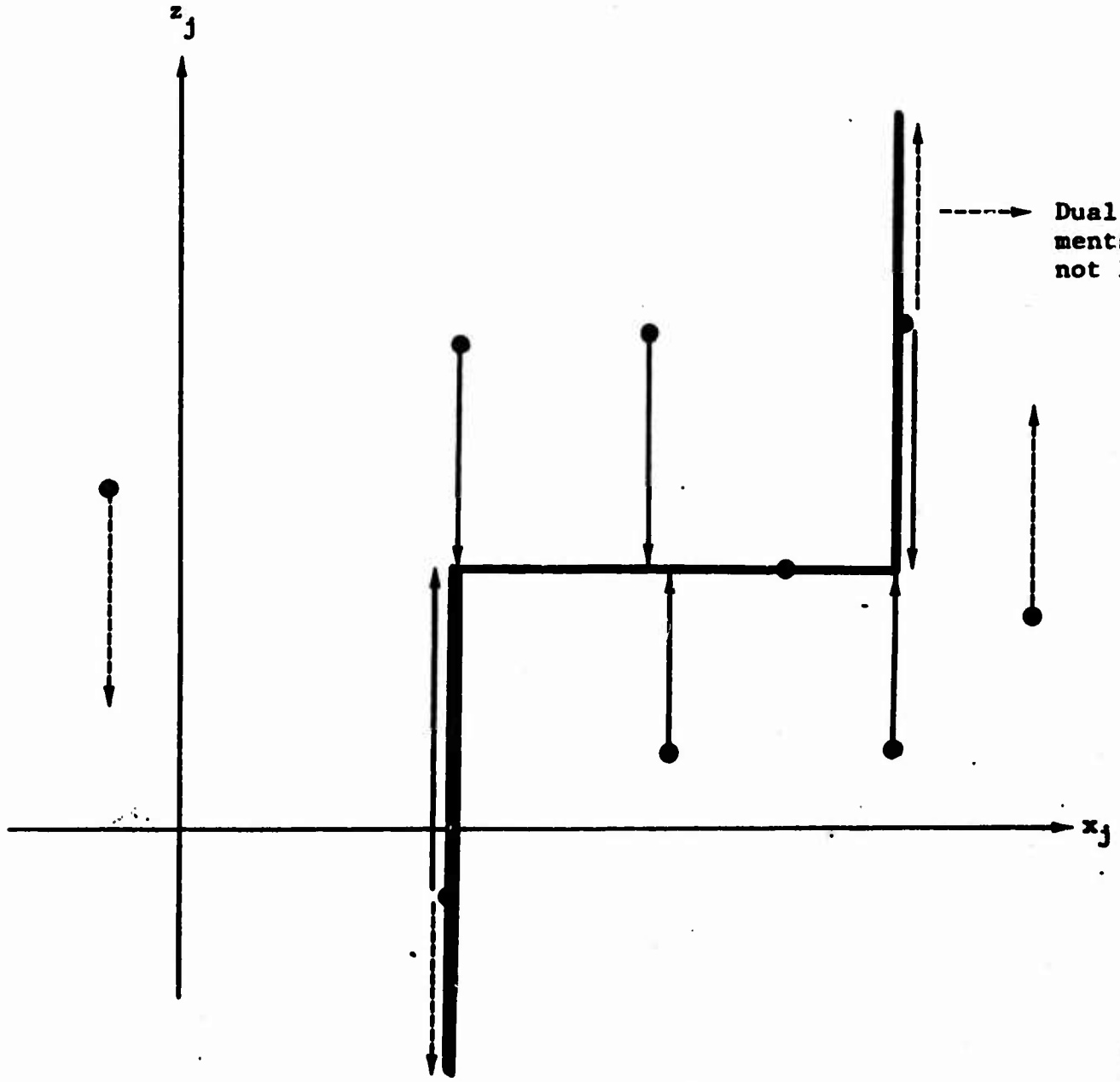


FIGURE 4.1: OPTIMALITY DIAGRAM FOR ACTIVITY j , SHOWING DIRECTION AND MAXIMAL DUAL DISPLACEMENTS ALLOWED FOR DIFFERENT STATES (●)

(4.6)
$$\begin{array}{llll} \text{(a)} & K \to B & \text{(c)} & L \to B \\ \text{(b)} & K^- \to B & \text{(d)} & L^+ \to B \end{array}$$

occurs. If s becomes conforming, then all the ζ_j^* may be zero, in which case θ^* can be set to zero. If the states referred to in (4.5) are nonexistent, and s is nonconforming, then $\theta^* = \infty$, and the dual solution is unbounded.

Of particular interest is the trajectory traced out by (x_s, z_s) as it is "worked upon" by the incremental linear program. Suppose first that $(x_s^o, z_s^o) \in U^-$, as shown in Figure 4.2; then the subroutine will maximize ξ_s. The result of this subroutine will be $\Delta^* = \xi_s^*$, creating a nonnegative displacement towards the right to (x_s^1, z_s^1), possibly all the way to state K, (or unbounded if $\kappa_s = \infty$). (For $z_s^o < c_s$, the first step might, of course, be stopped by entry into state L.)

Then calculation of the dual changes will change z_s by the amount $\theta^* \zeta_s^*$. But, from (2.9) and (3.2), if the horizontal segment does not reach to state K or L, then $\xi_s^* < \kappa_s$, and $\zeta_s^* = -1$. In other words, the dual change moves the trajectory *downwards* by a positive amount θ^* to the point (x_s^1, z_s^1). If $\theta^* = \infty$, then the trajectory moves downwards off the diagram, showing dual unboundedness (i.e., primal infeasibility--there no basis change at any price which will bring x_s up to ℓ_s).

The net result is that, after successive applications of the subroutine, the trajectory of (x_s^t, z_s^t) describes what Dennis [8] calls the *breakpoint curve*, a sequence of nonnegative horizontal segments and positive vertical segments which leads either to unboundedness or infeasibility, or to an intersection with one of the conforming "U" states. Similar remarks, "in reverse", apply to the trajectory traced out by some activity in U^+, for which the subroutine would minimize ξ_s. (See also Section 8 for further possibilities.)

Unselected non-conforming activities also follow a breakpoint curve at each application of (2.2) and (4.2) until conforming, or selected.

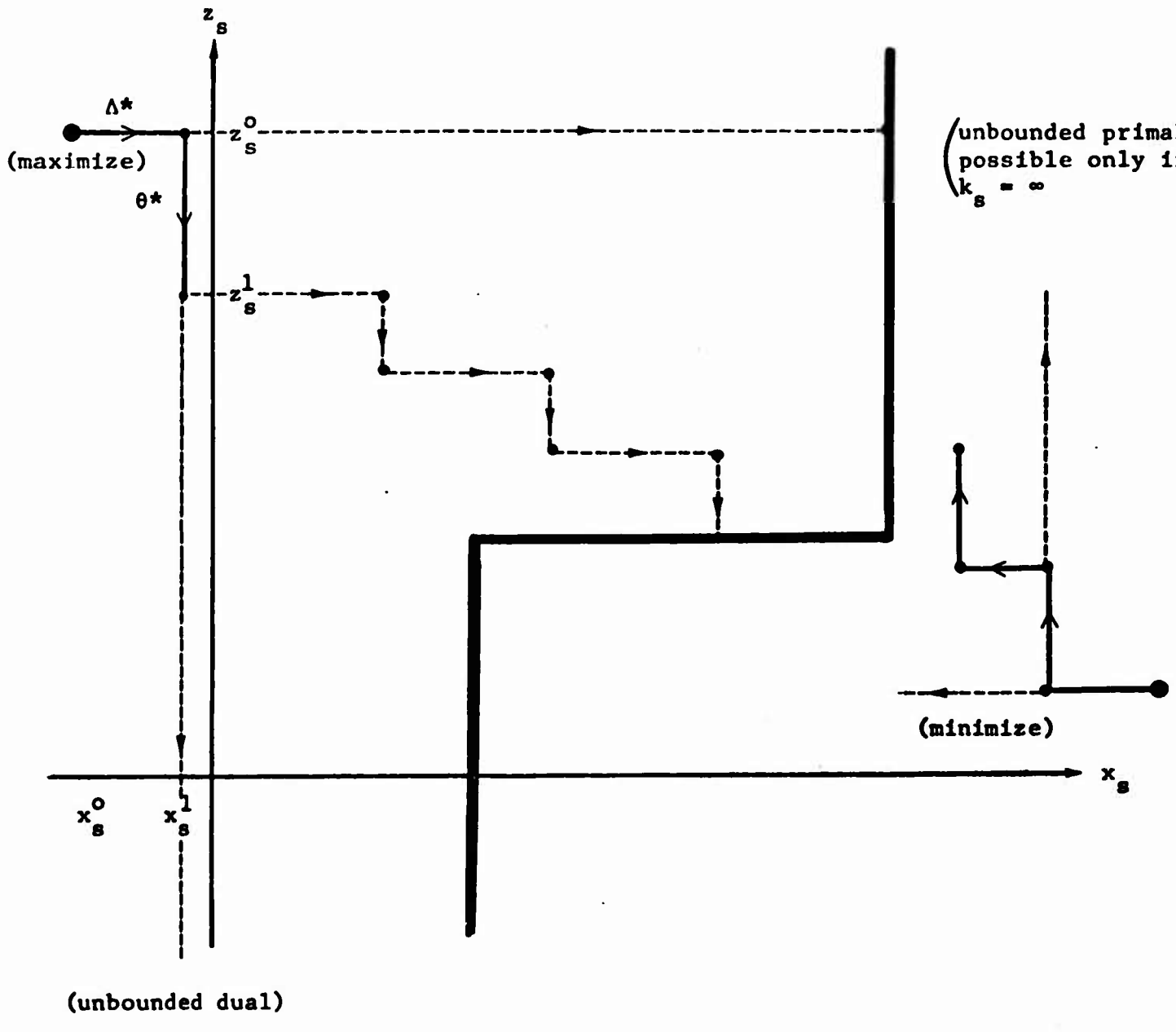


FIGURE 4.2: BREAKPOINT STEPPING THE SELECTED ACTIVITY AS A RESULT OF THE INCREMENTAL LINEAR PROGRAM

5. GENERAL OUTLINE OF THE ALGORITHM

The general outline of the algorithm should now be apparent. Starting with an arbitrary solution to the primal and dual, the states of each activity are identified, using (1.3) and Figure 1.1. An arbitrary nonconforming activity, s, is selected and worked upon, using the incremental subroutine. By following the breakpoint curve, Figure (4.2) either unboundedness or infeasibility is shown, or s is put in-kilter. Then another nonconforming activity is selected, and so on, until all activities are in U, and an optimal solution is obtained.

There are still several points to be cleared up, such as the finiteness of the procedure, both with the regard to the subroutine, and the main algorithm (Section 7). In addition, there are various options available at each iteration which can be used in devising various heuristic procedures; these will be discussed in Section 8 and when comparing the algorithm with others in Appendix D.

We now present the main algorithm.

6. COMPLEX: A COMPLEMENTARY SLACKNESS, OUT-OF-KILTER ALGORITHM FOR LINEAR PROGRAMMING

0. Select arbitrary values of $x_j = x_j^o$ $(j = 1,2, \ldots, n)$ satisfying (1.1) and $y_i = y_i^o$ $(i = 1,2, \ldots, m)$. Select an arbitrary set of m independent columns for the initial set $\mathcal{J} = \mathcal{J}^o$ (Appendix B). Set $t = 0$.

1. Identify current states $(x_j^t; z_j^t)$ of activities as $U^- = L^-$, B^-, or K^-; $U = L$, B, or K; $U^+ = L^+$, B^+, or K^+. If all $j \in U$, the current solution is optimal.

2. Otherwise, select an arbitrary nonconforming state s. Solve the *Incremental Linear Program*

$$\text{Maximize } \Delta = \sum_{j=1}^{n} \epsilon_j \xi_j$$

$$\sum_{j=1}^{n} a_{ij}\xi_j = 0 \qquad (i = 1,2, \ldots, n)$$

$$\lambda_j \le \xi_j \le \kappa_j \qquad (j = 1,2, \ldots, n) \qquad (6.1)$$

$$\text{Minimize } \sum_{j=1}^{n} (\kappa_j \upsilon_j - \lambda_j \omega_j)$$

$$\sum_{i=1}^{m} \eta_i a_{ij} - \upsilon_j + \omega_j = -\epsilon_j$$

$$\upsilon_j \ge 0 \,, \quad \omega_j \ge 0$$

$$\eta_i \text{ unrestricted}$$

with initial basis $\mathcal{J}^t$; starting solution $\xi_j = 0$ $(j = 1, \ldots, n)$;

(6.2)
$$\varepsilon_j = \begin{cases} 0 & j \neq s \\ +1 & j = s \in U^- \\ \text{or} & \\ -1 & j = s \in U^+ \end{cases}$$

and $\{\lambda_j, \kappa_j\}$ defined by (3.2).

3. If $\Delta^* = +\infty$, the primal problem (0.1) is unbounded.

4. Otherwise, set

(6.3)
$$x_j^{t+1} = x_j^t + \xi_j^* \qquad (j = 1, \ldots, n)$$

5. Find the step size, θ^*, from (4.5) and the related discussion.

6. If $\theta^* = \infty$, the primal problem (0.1) is infeasible.

7. Otherwise, set

(6.4)
$$y_i^{t+1} = y_i^t + \theta^* \eta_i^* , \qquad (i = 1,2, \ldots, m)$$

(6.5)
$$z_j^{t+1} = z_j^t + \theta^* \zeta_j^* , \qquad (j = 1,2, \ldots, n)$$

(retain the current optimal basis $\mathcal{B}^*$ as the starting basis $\mathcal{B}^{t+1}$ for the next iteration) and repeat Step 1.

7. CONVERGENCE AND FINITENESS

Assume temporarily that the current solution is "primal-feasible", i.e., $\ell_j \leq x_j \leq k_j$. By direct substitution, the difference between the primal and dual functionals is:

$$\mathcal{D} = \mathcal{C} - \mathcal{B} = \sum_{j=1}^{n} u_j(k_j - x_j) + \sum_{j=1}^{n} w_j(x_j - \ell_j) \tag{7.1}$$

which will be called the *total deviation* of the current solution.

In terms of the optimality diagrams, the total deviation is just *the sum of the individual activity deviations, which are the shaded areas shown in Figure 7.1.*

Furthermore, this result does not require k_j or ℓ_j to be finite, provided one uses the usual "transfinite algebra" in which the infinite bound is replaced by a number $\pm M$ which will be considered larger than any number to which it is compared during the calculations. Thus to a point like Ⓐ in Figure 7.1, an area $u_j(M - x_j)$ would be contributed to the total deviation $\mathcal{D}$. This is just the pricing-in term which would be added to force the point out of the "dual-infeasible" region above the line $z_j = c_j$.

A similar device can be used to make arbitrary points "primal-feasible" as well. For example, if the current $x_j > k_j$ or $< \ell_j$, we may consider that x_j is really an unbounded activity with piecewise linear cost structure

$$\text{cost of activity } j = \begin{cases} c_j\ell_j + M(\ell_j - x_j) & -\infty \leq x_j < \ell_j \\ c_j \cdot x_j & \ell_j \leq x_j \leq k_j \\ c_j \cdot k_j + M(x_j - k_j) & k_j < x_j \leq \infty \end{cases} \tag{7.2}$$

which gives the transfinite extensions to the optimality curves shown in Figure 7.2 (see also Chapter VI, [4]).

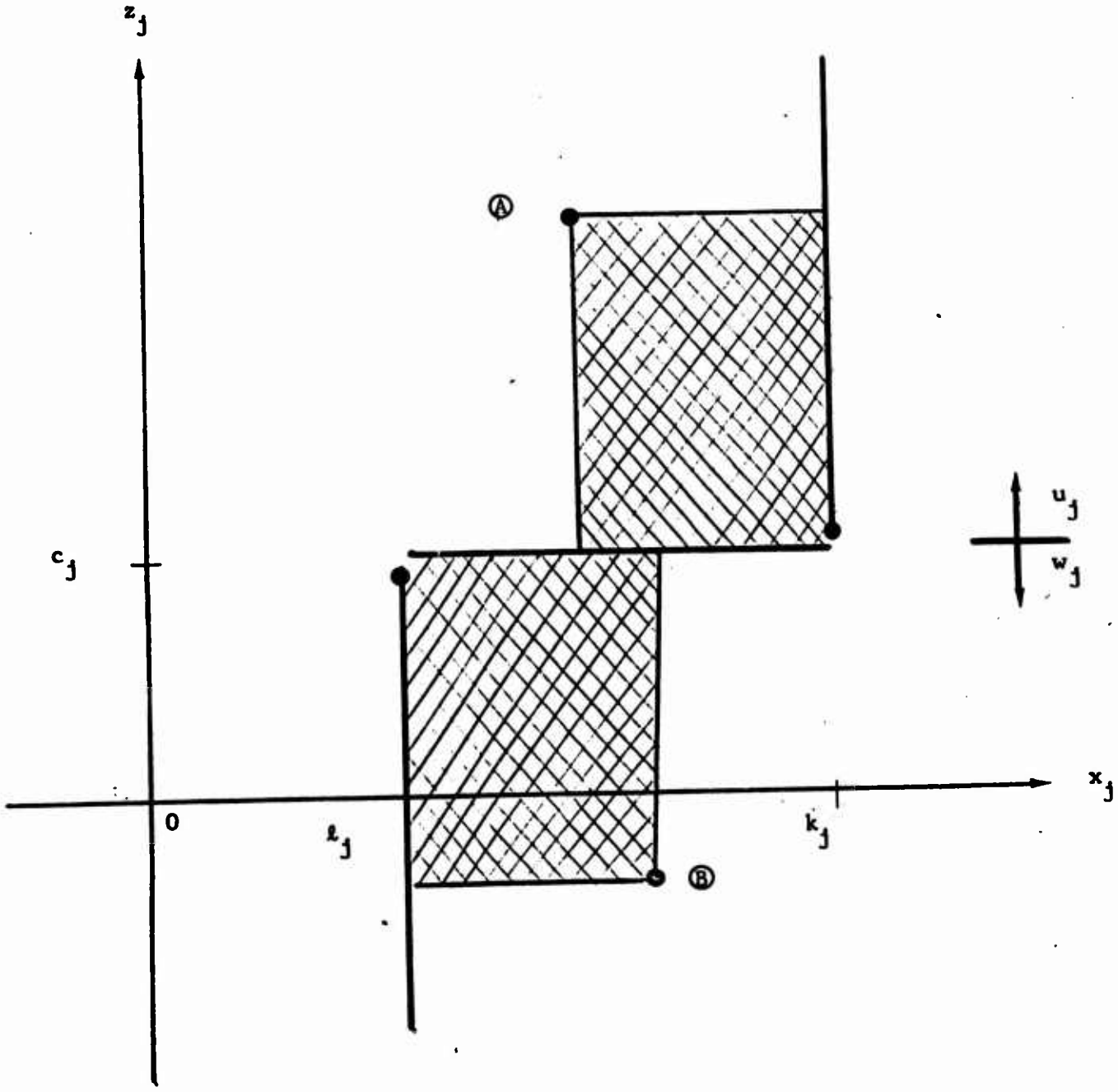


FIGURE 7.1: CONTRIBUTION TO THE TOTAL DEVIATION

$\mathcal{D} = \mathcal{C} - \mathcal{B}$ FROM "PRIMAL-FEASIBLE" $(x_j; z_j)$

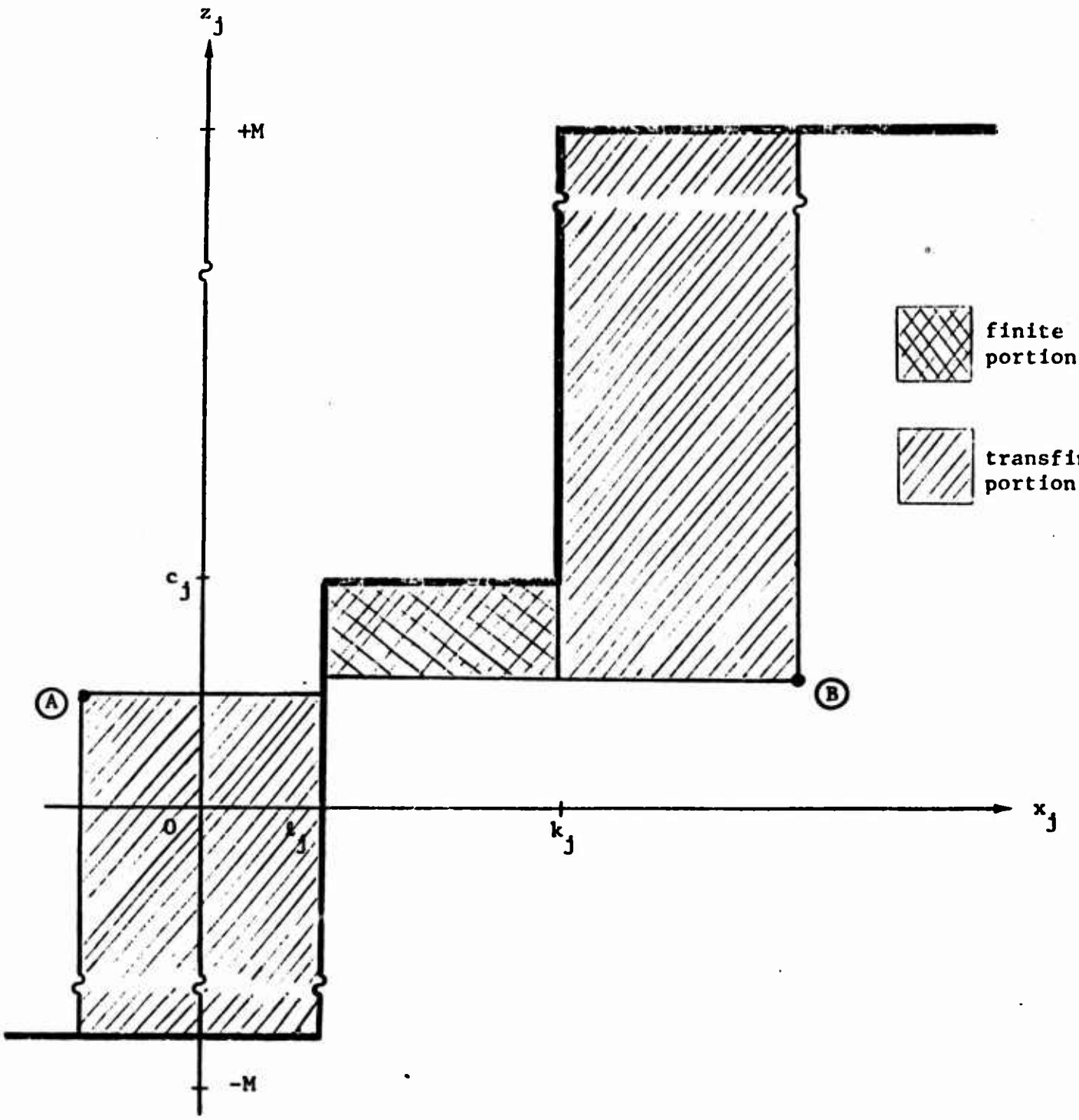


FIGURE 7.2: CONTRIBUTION TO THE TOTAL DEVIATION $\mathcal{D} = \mathcal{C} - \mathcal{B}$ FROM "PRIMAL-INFEASIBLE" $(x_j; z_j)$

Direct calculation of $\mathcal{D} = \mathcal{C} - \mathcal{B}$ for the extended problem (7.2) leads again to the conclusion that the area between the point $(x_j; z_j)$ and the corresponding diagram is the deviation due to activity j ; as shown in Figure 7.2, this deviation may be transfinite only (Ⓐ) , or may have both finite and transfinite regions (Ⓑ) .

Thus, as a given activity follows its breakpoint curve, as shown in Figure 7.3, it follows from the algorithm that:

(7.3) *Every nonzero horizontal or vertical displacement permitted by the algorithm gives a finite decrease to the total deviation* $\mathcal{D} = \mathcal{C} - \mathcal{B}$, *equal to the decrease in shaded area between the points* $(x_j; z_j)$ *and the optimality curves, summed over all activities.*

In classical terminology, the displacement from I to II in Figure 7.3 makes activity j "primal-feasible", and from II to III, "optimal" (if k_j was $+\infty$, all vertical displacements reduced "dual-infeasibility", as well); however, in our extended definitions, all horizontal displacements decrease $\mathcal{C}$, and all vertical displacements increase $\mathcal{B}$.

Since the only displacements allowed decrease $\mathcal{D}$, and since each non-conforming activity is in the subroutine until it becomes conforming, it is clear that the algorithm converges.

The only possible source of degeneracy occurs in the incremental subroutine, where cycling can be avoided by the usual perturbation or lexicographic techniques[4]. Even if the maximal value of Δ is zero (2.8), the decrease in $\mathcal{D}$ will be positive, and a new basis will be selected. Thus the algorithm is finite.

An alternate "finite" proof that an infinite number of steps with θ^* finite

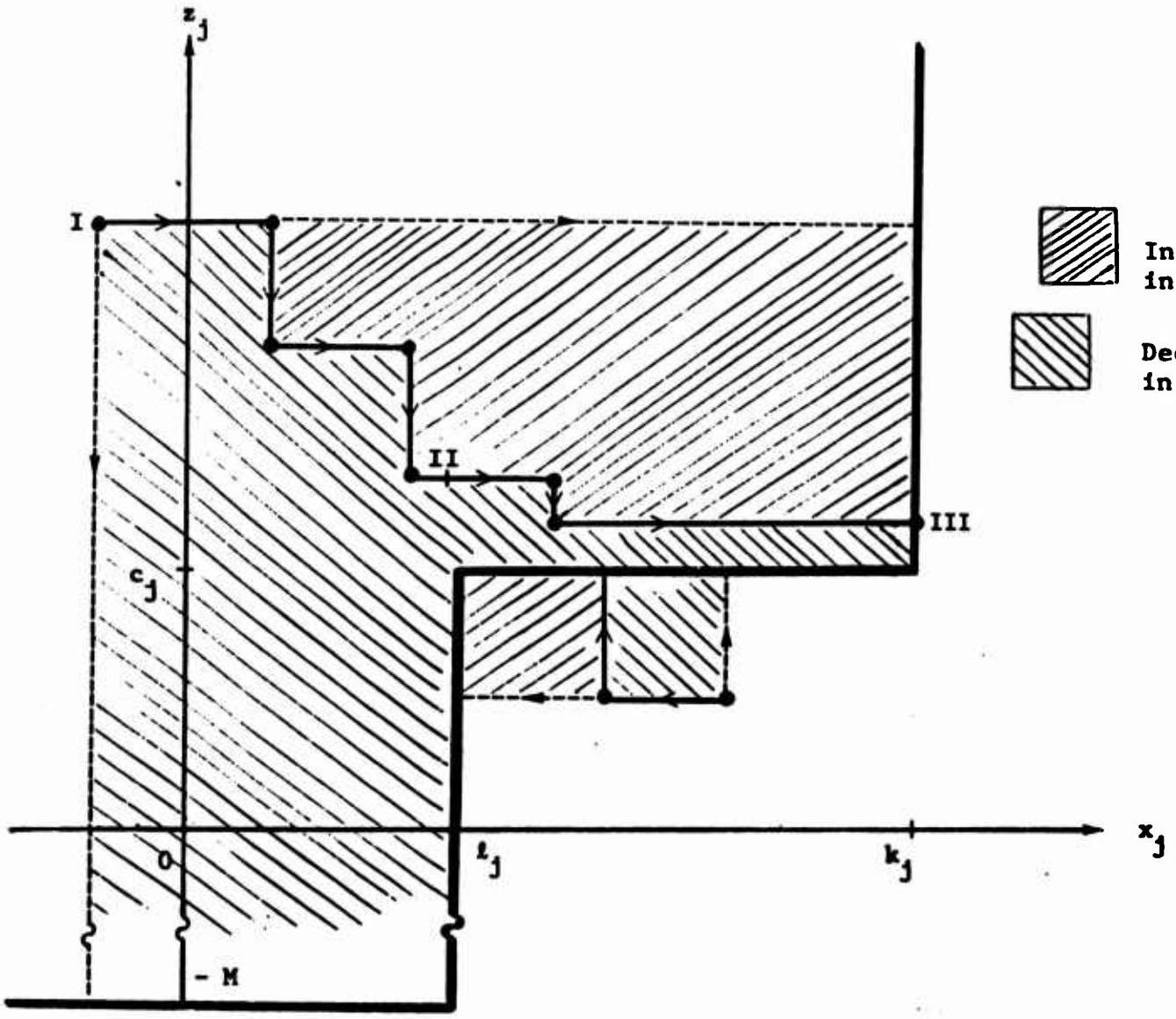


FIGURE 7.3: DECREASE IN THE TOTAL DEVIATION $\mathcal{D} = \mathcal{C} - \mathcal{B}$ AS ACTIVITY j BECOMES CONFORMING

and with $\Delta^* = 0$ cannot occur is as follows:

(a) Since $\Delta^* = 0$, no changes of the type

(i) $K^+ \to K$

(ii) $K^+ \to K$

(iii) $L^- \to L$

(iv) $L^+ \to L$

(v) $(j \in B)$ $\quad (x_j = \ell_j) \overset{\to}{\leftarrow} (x_j > \ell_j)$

(vi) $(j \in B)$ $\quad (x_j < k_j) \overset{\to}{\leftarrow} (x_j = k_j)$

(7.4)

can occur.

(b) If an activity moves from $L \to B$ or $K \to B$ during one dual change, its index will enter the basic set J of the subroutine and remain there as long as $\Delta^* = 0$.

(c) Otherwise, θ^* is determined by a transition $L^+ \to B$ or $K^- \to B$ decreasing the number of activities which enter into (4.5).

(d) Since there are a finite number of activities, there can be only a finite number of dual changes until either some Δ^* is finite ("breakthrough"), or the set in (4.5) is empty (unbounded dual).

8. ALTERNATIVES FOR THE INCREMENTAL PROGRAM SUBROUTINE

The incremental linear program (2.2) can, of course, be solved by any method available; a fairly compact tableau method which utilizes its homogeneous, double-bounded structure is given in Appendix A. In this section, we indicate several alternative approaches which may be used.

A. Working on Several Nonconforming Activities at the Same Time

If, in fact, a general simplex tableau procedure, as outlined in Appendix A, is being used to solve the incremental program, then one may wish to do more than maximize (or minimize) the selected nonconforming activity. In particular, one may work on *all* such activities at once, by setting

$$\epsilon_j = \begin{cases} +1 & \text{if} \quad j \in U^- \\ -1 & \text{if} \quad j \in U^+ \\ 0 & \text{otherwise} \end{cases} \tag{8.1}$$

instead of (A.1) and (6.2). This does not complicate the subroutine outlined in Appendix A and *may* make several activities conforming in one application (with possibly more pivot steps) of the subroutine.

Or, one may select *some* of the nonconforming activities to work on--for example, the activities with transfinite deviations may be worked on first, as in the usual "Phase I" procedures (Appendix D).

Finally, the magnitude of the coefficients ϵ_j is immaterial to convergence of the algorithm, and one may choose to put more or less pressure on certain non-conforming activities, based on some heuristic choice; this is the basis for most proposals which combine "Phase I" and "Phase II".

B. Primal-Freezing Nonconforming Activities for Single-Step Subroutines

A possibility in the other direction is to make the incremental program as

simple as possible. One way to do this is to "freeze" all nonconforming variables, other than the one selected, at their current values by setting

(8.2) $$\lambda_j = \kappa_j = 0 \qquad j \notin U + \{s\}$$

instead of following (3.2). λ_s or κ_s are set as usual.

In this way, as the subroutine (Appendix A) attempts to increase (or decrease) ξ_s, only two bounded possibilities occur:

(8.3)

(a) ξ_s reaches one of its own nonzero bounds; the subroutine terminates with s conforming and the current basis $\mathcal{J}$ unchanged (but with possibly different ξ_j, $j \in \mathcal{J}$); all $\zeta_j^* + \varepsilon_j$ are unchanged (equal to ε_j).

or

(b) One (or more) basic ξ_ℓ reaches a bound; s replaces ℓ in the basis by making a pivot on some a_{is} (A.6). *The subroutine terminates after one pivot* with $\zeta_\ell^* + \varepsilon_\ell^* \left(\gtrless\right) 0$, and all other $\zeta_j^* + \varepsilon_j = 0$, $j \in \mathcal{J} + \{s\}$.

(The usual remarks about ties apply.)

In this simpler, but more restricted procedure, it is clear that no control is maintained over the sign of ζ_j^*, $j \notin U + \{s\}$. Thus, *other nonconforming activities may become more so* (i.e., their deviations may increase) at the dual-changing step which follows. This gives some theoretical problems in convergence, but in most cases, one can show that one of the functionals is moving in the correct direction.

If the resulting value of ξ_s^* from (8.3) were then such that x_s were "primal-feasible", then it would be desirable to choose θ without regard to the other nonconforming activities. This would then make s *conforming in a single pass through the algorithm*, although other activities might "overshoot"--i.e., *conforming ones might become nonconforming* (see D below).

The above procedure is used in the primal simplex algorithm. If several variables are brought in at once, the procedure is called "block-pivoting".

C. Dual-Freezing Certain Nonconforming Activities

In a similar way, one can dual-freeze up to m nonconforming variables ζ_j^* by requiring that their indices be in any basis $\mathcal{J}$ of the incremental subroutine. This restriction then means that the bounds λ_j and κ_j (3.2) for these variables *cannot be effective*, and the corresponding ξ_j^* may have arbitrary sign; thus these other nonconforming variables may have increasing deviations.

If an attempt is made to make activity s conforming in one step, this may drive some x_j, $j \in B$ nonconforming. Thus, this possibility is usually followed *in reverse*; i.e., some ξ_j, $j \in B$ ($\zeta_j = 0$) is moved towards its bound until some positively or negatively priced ξ_s reaches the appropriate bound. This "row-pivoting" is the procedure used in the dual simplex algorithm (Appendix D).

D. Overshooting Conforming Limits

In general, the limits on the ξ_j and $\theta\zeta_j$, $j \notin U$, have been chosen so that:

(a) no conforming activity passes through a conforming state and then becomes nonconforming again;

(b) no previously conforming activity becomes nonconforming.

As we have seen above, however, when working on a particular activity, or set of activities, it may be desirable to get this activity conforming "at all

costs". This may conceptually be very bad if violating (3.2) or (4.5) makes more activities nonconforming. However, in certain special cases, we may be able to argue convergence on one of the functionals.

E. Special Structure Models

Actually, the incremental subroutine, as we see it, is a method for choosing the *direction* of the vector $\{\xi_j\}$ which will maximize the rate of increase of Δ, subject to $\sum a_{ij}\xi_j = 0$, with certain directions "frozen"; the actual maximum displacement in either primal or dual is of secondary importance.

In problems of special structure, it may be possible to follow through the effect of changing one variable on the other variables explicitly; then the various operations of "pivoting" can be carried out in sequential form, without continual reduction of the matrix to get the current trade-off coefficients α_{ij}.

The most common example of this kind occurs in network flow models, where the allowed changes in the $\{\xi_j\}$ correspond to an incremental increase in arc flows around a loop including arc s. (Section 10.)

F. Dual-Stepping

Nothing in the algorithm should be construed so as to give a special place to the primal problem. One can just as well define the $\{\eta_i\}$ as the *absolute* displacements of the $\{y_i\}$, and work on a selected η_s through a homogeneous dual in the incremental subroutine. For example, the pivoting procedure may be clearer in the transposed matrix (a_{ij}).

In this case, nonnegative vertical steps in Figure 4.2 are taken first, followed by positive horizontal displacements, since all degeneracy (at the corner points of Figure 1.1) arise in the subroutine. Thus, "row pivots" become the natural changes, and "column pivots" would require looking ahead to the "primal-" changing step (6.4) and (6.5).

G. Bound-Tightening

When a given activity is made more conforming, it is possible some other nonconforming variable may move coherently. If the incremental subroutine makes several basis changes, it may be worthwhile to "tighten" the bounds on these other variables as the pivoting progresses to prevent them from "slipping back". Actually, except for didactic examples, this possibility is quite rare.

H. STARTING SOLUTIONS

In many problems, special starting solutions, basic or not, may be available. If these are felt to be "reasonable", or near optimal, they certainly should be used in place of a completely arbitrary solution. On the other hand, if one had previously found a basic set J^o, and the related inverse basis in (a_{ij}), then one should use the corresponding basic solution, feasible or not, solely in the interest of efficiency.

I. ARGUMENTS FOR EFFICIENCY

It should be clear from the discussion of this Section and Appendix D that any prior arguments for efficiency of a certain heuristic, particularly those based on whether one is moving "inside", "outside" or "on" a certain convex polytope, are doomed to failure.

The COMPLEX algorithm takes *any* starting solution, basic or not, feasible or not, and converts it into an extreme point by adding artificial bounds (2.4). Thus all starting solutions "look alike", in a certain sense, and progress in the same manner as a basic feasible solution would move over the original polytope. One would have to make extensive *numerical* trials for special classes of problems in order to clearly demonstrate the superiority of one heuristic over another. Most such experiments have concentrated on how to select a nonconforming activity when using the heuristic described in Section 8B above, and starting with basic feasible solutions.

9. SYMMETRIC FORMULATIONS

The problem (0.1) (0.2) has been stated as an equality primal with doubly-bounded variables, since this is often the formulation in real problems. On the other hand, certain models, such as two-person games, look more natural in a symmetric primal-dual format; this approach is often favored for aesthetic reasons, as well. In this section, we modify the algorithm of Section 6 to a symmetric form.

Consider:

$$\text{Minimize } C = \sum_{j=1}^{n} c_j x_j$$

(9.1)

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i \qquad (i = 1,2, \ldots, m)$$

$$x_j \geq 0 \qquad (j = 1,2, \ldots, n)$$

$$\text{Maximize } B = \sum_{i=1}^{n} b_i y_i$$

(9.2)

$$y_i \geq 0$$

$$\sum_{i=1}^{m} y_i a_{ij} \leq c_j$$

By defining nonnegative slack variables:

(9.3) $$r_i = \sum_{j=1}^{n} a_{ij} x_j - b_i \geq 0 \qquad (i = 1,2, \ldots, m)$$

(9.4) $$s_j = c_j - \sum_{i=1}^{m} y_i a_{ij} \geq 0 \qquad (j = 1,2, \ldots, n)$$

the constraints in (9.1) and (9.2) are changed to equalities; since they will remain equalities during the algorithm, they are henceforth ignored, and all attention is focussed on the *extended variables:*

$$\chi_k = \begin{cases} x_k & (k = 1,2, \ldots, n) \\ r_{k-n} & (k = n+1, n+2, \ldots, n+m) \end{cases} \tag{9.5}$$

$$\gamma_k = \begin{cases} s_k & (k = 1,2, \ldots, n) \\ y_{k-n} & (k = n+1, n+2, \ldots, n+m) \end{cases} \tag{9.6}$$

where k runs over the range $(1,2, \ldots, n;n+1, n+2, \ldots, n+m)$ to take in the appropriate real or slack variables. The *extended constraint matrix* of (9.1) consists of (a_{ij}) augmented by an $m \times m$ negative identity matrix:

$$(a_{ij}) = ((a_{ij}) \vdots -I) \tag{9.7}$$

The restatement of the Optimality Principle (1.5) is:

Optimality Principle

A feasible solution of nonnegative values $\{\chi_k; \gamma_k\}$ *is optimal if and only if*

$$\chi_k \cdot \gamma_k = 0 \tag{9.8}$$

for all $k = 1,2, \ldots, m+n$.

The optimality diagrams corresponding to (9.7) are shown in Figure 9.1. Note that this figure is reversed and normalized from Figure 1.1. Thus, the breakpoint trajectories will have reversed "dual" changes, increasing (decreasing) from left to right (right to left). We keep the same states L^- , L , L^+ , B^- , B , and K^- as before for the current *extended state* $\{\chi_k; \gamma_k\}$.

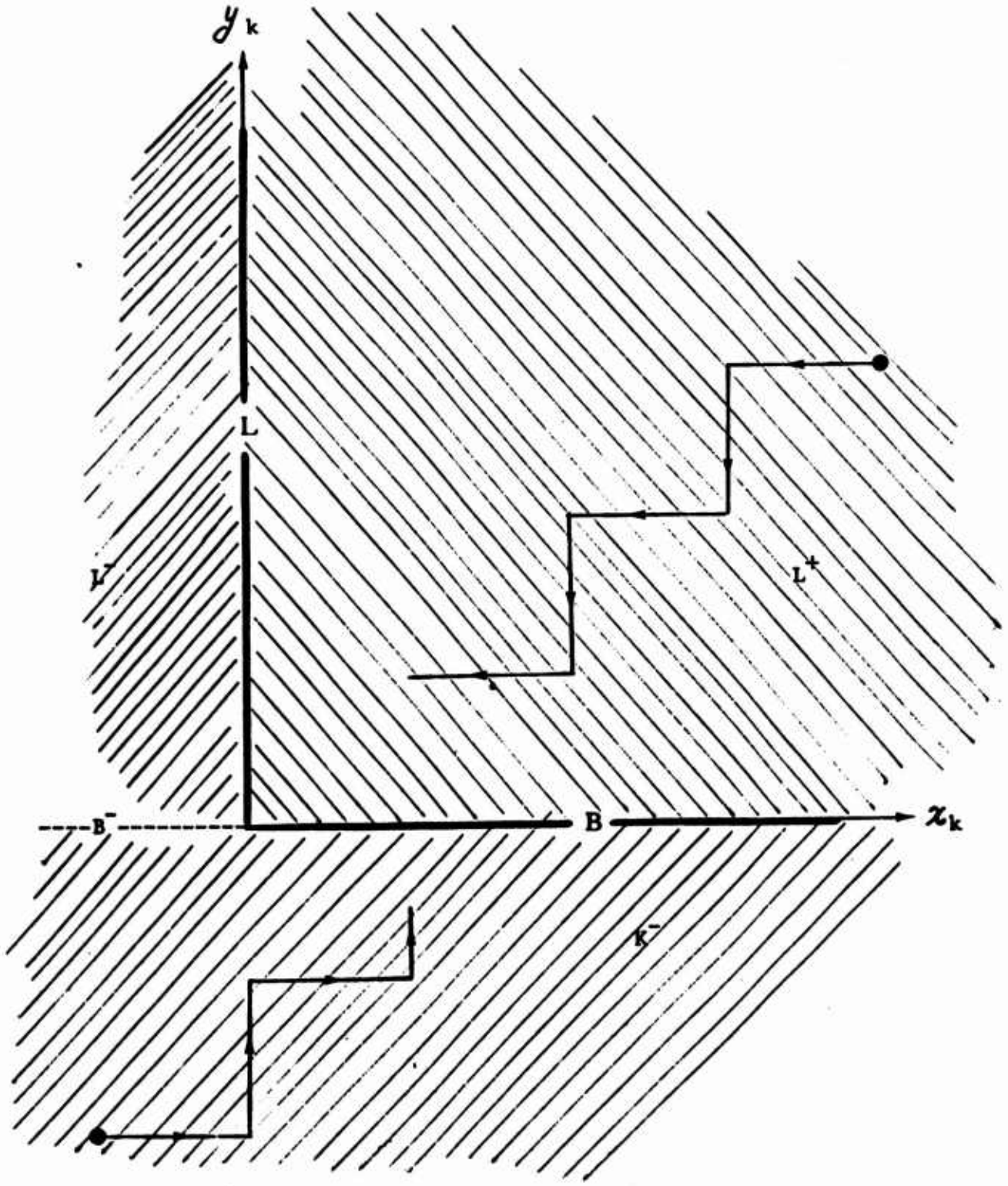


FIGURE 9.1: OPTIMALITY DIAGRAM FOR SYMMETRIC FORM

For completeness, we restate the algorithm of Section 6 in symmetric form; some details on the extended tableau are given in Appendix C. No proofs of the algorithm need be given, since the extended problem is exactly in the form (0.1) and (0.2). However, some "complementary pivot" interpretations are given in subsection B.

A. Symmetric Form of the Complex Algorithm

0. Select arbitrary values of $x_j = x_j^o$ $(j = 1,2, \ldots, n)$ and $y_i = y_i^o$ $(i = 1,2, \ldots, m)$ and use (9.3)(9.4) to calculate the remaining components of $\{\mathcal{X}_j^o\}$ and $\{\mathcal{Y}_k^o\}$.
Select an arbitrary set of m independent columns $\mathcal{A}^o$, in the extended constraint matrix, (a_{ij}) (say, the negative identity matrix of the last m columns).
Set $t = 0$.

1. Identify the current states $\left(\mathcal{X}_k^t; \mathcal{Y}_k^t\right)$ of all variables as $U^- = L^-$, B^-, or K^-; $U = L$ or B; $U^+ = L^+$. If all $k \in U$, the current solution is optimal.

2. Otherwise, solve the *Incremental Linear Program:*

$$\text{Maximize } \Delta = \sum_{k=1}^{m+n} \varepsilon_k \xi_k$$

$$\sum_{k=1}^{m+n} a_{ik}\xi_k = 0 \qquad (i = 1,2, \ldots, m) \tag{9.9}$$

$$\lambda_k \leq \xi_k \leq \kappa_k \qquad (k = 1,2, \ldots, m+n)$$

$$\text{Minimize} \quad \sum_{k=1}^{m+n} (\kappa_k v_k - \lambda_k \omega_k)$$

(9.10)

$$\sum_{i=1}^{m} \eta_i a_{ik} - v_k + \omega_k = -\varepsilon_k \qquad (i = 1,2, \ldots, m)$$

$$v_k \geq 0 \; ; \; \omega_k \geq 0 \qquad (k = 1,2, \ldots, m+n)$$

$$\eta_i \text{ unrestricted}$$

with initial basis $\mathcal{J}^t$, and starting solution $\xi_k = 0$ $(k = 1,2, \ldots, m+n)$.

The coefficients in the functional have arbitrary value and sign:

(9.11)

$$\varepsilon_k \begin{cases} \equiv 0 & k \in U \\ \geq 0 & k \in U^- \\ \leq 0 & k \in U^+ \end{cases}$$

selected to work on one or several nonconforming variables at the same time

The bounds are:

(9.12)

k	λ_k	κ_k
L^-	0	$-\chi_k^t$
B^-	0	∞
K^-	0	∞
L	0	0
B	$-\chi_k^t$	∞
L^+	$-\chi_k^t$	0

3. If $\Delta^* = +\infty$, the primal problem (9.1) is unbounded.

4. Otherwise, set:

(9.13) $$\gamma_k^{t+1} = \gamma_k^t + \xi_k^* \qquad (k = 1,2, \ldots, m+n)$$

5. Find the step size θ^* from

(9.14) $$\theta^* = \min_k \left(y_k^t \Big/ \sum_{i=1}^m \eta_i^* a_{ik} \right)$$

where only indices k are allowed for which: $\gamma_k^{t+1} \geq 0$; the denominator is nonzero; and the numerator and denominator are of the same sign.

6. If the set of indices in (9.14) is empty, $\theta^* = \infty$, and the primal problem (9.1) is infeasible.

7. Otherwise, set:

(9.15) $$y_k^{t+1} = y_k^t - \theta^* \sum_{i=1}^m \eta_i^* a_{ik} \qquad (k = 1,2, \ldots, m+n)$$

and repeat Step 1.

B. Complementary Pivot Methods

Appendix C points out how certain of the basis changes in the extended matrix (a_{ij}) can be interpreted as "dual pivots", in the sense of the row operations of the dual simplex method (Appendix D.A). This symmetrization is formalized in the *complementary pivot methods* of Cottle [3], Dantzig [6], and Lemke [12], which were developed for a larger class of problems (Section 13).

Instead of the extended variables (9.5)(9.6), attention is focussed on the variables:

(9.16) $$z_k = \begin{cases} x_k & (k = 1,2, \ldots, n) \\ y_{k-n} & (k = n+1, n+2, \ldots, n+m) \end{cases}$$

(9.17) $$w_k = \begin{cases} s_k & (k = 1,2, \ldots, n) \\ r_{k-n} & (k = n+1, n+2, \ldots, n+m) \end{cases}$$

This has the effect of reversing the axes on the last m optimality diagrams of Figure 9.1, thus making the "real" dual variables y_i abscissae, together with the "real" primal variables x_j. The problem (9.1) (9.2) is also usually stated as a combined $(m+n) \times (m+n)$ primal-dual problem, together with feasibility requirements which are identical with the optimality requirements (9.8).

In terms of our model, complementary pivot theory stresses the case when exactly *one* pair of *complementary variables*, say $(z_h; w_h)$ in (9.16) (9.17) is nonconforming; this implies exactly one other pair is at the LB corner point, say $z_e = w_e = 0$ (possibly more than one if there is degeneracy). This point e is then moved into either L or B in a manner to reduce the nonconformity of the variable h, some other variable ℓ then moving into its corner. If ℓ moves into the corner on B, it leaves on L at the next iteration, and vice versa; this is what is meant by complementary pivoting, and is a natural observation from the incremental subroutine of Appendix C. The procedure, of course, terminates, when variable h becomes conforming.

Actually, starting with just one nonconforming solution pair $(z_h; w_h)$ is quite difficult, in general, and the only starting solution methods proposed [5] seem to require introduction of artificial variables. Following the effect of this proposal through in terms of Figure 9.1 reveals:

(a) All initial and subsequent solutions are in L, B, or B^-, or $\left(\gamma_k < 0 \text{ and } \chi_k = 0\right)$.

(b) The nonconforming point(s) which is(are) currently farthest away (in a linear sense) are worked upon.

(c) Primal- and dual-freezing are used as needed to keep all solutions in the above states.

These special rules then reduce the complementary pivot procedure to a combination of the primal-simplex and dual-simplex procedures, that is, a composite method (Appendix D.I).

However, we have previously seen (2.7) that the *incremental program*, (2.2) or (9.9), is already in complementary pivot form, since only $\xi_s = 0$ is nonconforming at the beginning of the incremental subroutine. Thus, the COMPLEX approach essentially reduces *every initial solution* to a related complementary pivot problem. In addition, the nuisance problems of infeasibility and unboundedness are handled separately.

C. A Doubly Double-Bounded Symmetric Problem

As an ultimate symmetric variant, the reader may wish to try rewriting the algorithm of subsection A for the following *doubly double-bounded symmetric problem:*

$$\text{Min}\,\mathcal{C} = \sum_{j=1}^{n} c_j x_j - \sum_{i=1}^{m} [d_i \max(0,r_i) - e_i \max(0,-r_i)]$$

$$\sum_{j=1}^{n} a_{ij} x_j - r_i = b_i \qquad (i = 1,2, \ldots, m) \tag{9.18}$$

$$\ell_j \le x_j \le k_j \qquad (j = 1,2, \ldots, n)$$

$$\text{Max}\,\mathcal{D} = \sum_{i=1}^{m} b_i y_i + \sum_{j=1}^{n} [\ell_j \max(0,s_j) - k_j \max(0,-s_j)]$$

$$d_i \le y_i \le e_i \tag{9.19}$$

$$\sum_{i=1}^{m} y_i a_{ij} + s_j = c_j$$

whose optimality diagrams are given in Figure 9.2 a and b.

D. General Piecewise-Linear Convex Costs

It is also appropriate to remark that the COMPLEX algorithm can easily be extended to general piecewise-linear convex costs. The resulting optimality diagrams would then have many horizontal and vertical segments; only a few details in the algorithm would need to be changed. (See also Section 12 and [8].)

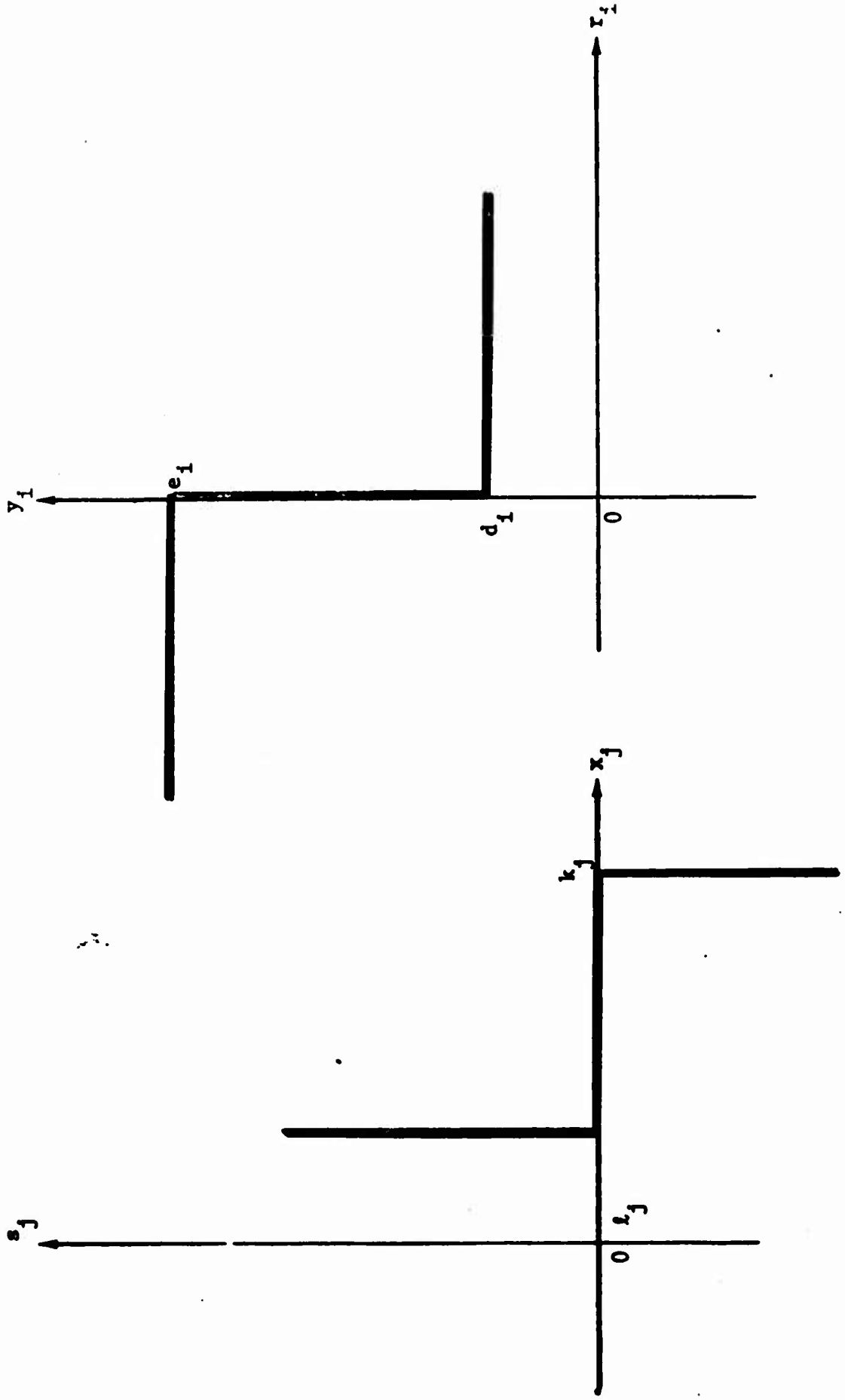


FIGURE 9.2: OPTIMALITY DIAGRAMS FOR DOUBLY DOUBLE-BOUNDED SYMMETRIC PROBLEM

10. NETWORK FLOW MODELS

A special class of problems of great practical interest are the network flow models [9,10,14] for which the constraint matrix is the node-arc incidence matrix, giving either Kirchoff's Current or Voltage Laws in the primal or dual.

For these models, the homogeneous incremental program takes on the following simple form:

(10.1)

(a) Find a loop (cycle) of special arcs in the network that can take an incremental amount of flow;

or

(b) Establish a set of potentials on each node, so that the sum of potential differences around any loop of special arcs is zero.

Either or both of these steps is handled in netowrk problems by means of a simple *labeling* technique, which is merely a way of "unraveling" the effects of a pivot change; by changing the flow variables around a loop (adding an incremental *circulation flow*), one increases or decreases only the variables necessary to keep the flow conservation laws satisfied. (If the problem is stated in single source-sink form, the pivoting procedure may find a flow-*augmenting* path.) As usual, determining the actual bound on the increment of flow, or determining the changes in potential to "break down" a new loop, are secondary calculations which can be performed independently. Thus, the calculations proceed from a basic solution (a *tree* of arcs), adding a new vector (a *cotree* arc which forms a unique loop with the tree), through a pivot change (determining an arc in the loop to be removed), to the new basis (another tree).

Viewed in terms of the optimality diagram and the algorithm of Section 6, the various methods can be viewed as follows:

(a) The *stepping-stone* methods are primal simplex algorithms which always maintain basic primal-feasible solutions as shown in Figure D.1. A set of potentials, placed on the basic tree, makes a cotree arc nonconforming. Then, flow is rerouted to determine a new basic solution, and so on until optimal.

(b) The *Hungarian-Ford-Fulkerson* methods were responsible for the development of the general *primal-dual method* for linear programs discussed in Appendix D and F. Usually all non-conformity is placed on the *return* arc from sink to source, which starts with zero flow. Thus, incrementally least-cost flow-augmenting paths are sought until all flow is allocated.

(c) The *out-of-kilter* method is exactly the algorithm of Section 6.

Actually, after the important early papers of Ford and Fulkerson see [9,14] for references) the state-of-the-art was such that many people realized that the complementary slackness conditions were the key to dealing with arbitrary initial condition (Appendix A of [10]). In fact, an independent development of the out-of-kilter approach may be found in the 1958 thesis of J. B. Dennis [8]. The emphasis is on electrical analogues, and *elementary activities* (Section 12), but his "black-box" approach, and "breakpoint-tracing" are identical to our notion of a selected activity, and sequential primal-dual changes needed to force this activity to be conforming. This important paper treats general programming problems in this same light (Sections 12 and 13).

One can also extend the simplified pivoting of ordinary network flows to problems where there are arbitrary *multipliers* of flow on each arc [10a], giving arbitrary coefficients in the node-arc incidence matrix. However, here the appropriate basis is not a tree, and the appropriate "unwound" form of a pivot step is to find a *flow-absorbing loop* in the network. Extensions to multi-commodity flows have also been suggested (see, for example, [10b]), but here the efficiency of the algorithms is less satisfactory.

11. THE SIMPLEX METHOD AND ITS HEURISTICS

By this point, our central thesis should be evident:

all the extreme point methods of linear programming are

equivalent to the original simplex method,

modified by the following heuristic choices:

(a) How is the problem formulated--inequalities, equalities, symmetric form, self-dual, upper or lower bounds, etc.?

(b) What initial solution, if any, is available for the primal and/or dual?

(c) Is the initial solution basic, or is a natural starting basis available?

(d) Which group of nonconforming variables is to be worked on first? Is any subset to be worked on simultaneously, or will they be handled individually? What criterion of nonconformity selects among these?

(e) What is the attitude towards the other nonconforming variables? Are they to be bounded towards conformity? Primal- or dual-frozen? Ignored?

(f) Is a general simplex (pivot) subroutine to be used, or a special algorithm? Will it do primal steps first, or dual steps?

(g) Are single-pivot changes to conforming states to be made? How much primal- or dual-change overshooting is to be allowed?

The primary advantage of the algorithm presented here is a pedagogical one, that different variants are easily presented, and compared. Also, the problem does not need to be forced into any artificial format; construction of the appropriate optimality diagram always "reminds" one of the appropriate procedure to be followed.

Finally, the COMPLEX algorithm presents a general framework within which new heuristics can be explored; certainly the basic similarity of the various breakpoint curves suggests that questions of relative efficiency must be decided computationally, and not on the basis of whether a solution stays feasible in a certain sense, or always makes some activity conforming at each step. In this way, we distinguish between the *theory* and the *art* of optimization.

12. BREAKPOINT-TRACING ELEMENTARY ACTIVITIES AS BUILDING BLOCKS AND THE BREAKPOINT-THEORY ALGORITHMS OF J. B. DENNIS

In this section we will consider the "construction" of an arbitrary linear program from certain *elementary activities*. In this way, we hope to shed some light on the pioneering work of J. B. Dennis [8].

As one analyzes the mechanism of the simplex algorithms, it becomes clear that there are only a certain number of elementary operations performed, and that each regular activity could be thought of as a composite of simpler elements. Typically, we have: a "pricing" mechanism which turns an activity "on" or "off"; a primal (in)equality; and a dual (in)equality.

The three canonical elementary activities might be:

(a) a no-cost *switching* variable $x_j^{(s)}$, which turns other activities "off" and "on" to arbitrary levels, whenever the pricing is negative, or tries to go positive. (Figure 12.1 a)

$$x_j^{(s)} \geq 0 \; ; \; c_j^{(s)} = 0 \; , \; z_j^{(s)} \leq 0 \tag{12.1}$$

(b) an unbounded *constant-cost rate* activity $x_j^{(c)}$, which "costs" or "profits", as the activity level is positive or negative. (Figure 12.1 b)

$$-\infty \leq x_j^{(c)} \leq \infty \; ; \; c_j^{(c)} = c_j \; , \; z_j^{(c)} = c$$

(c) a *constant-level* activity, $x_j^{(k)}$, independent of pricing and costs. (Figure 12.1 c)

$$x_j^{(k)} = k_j \; ; \; (c_j^{(k)} = 0), \; z_j^{(k)} \text{ unrestricted.}$$

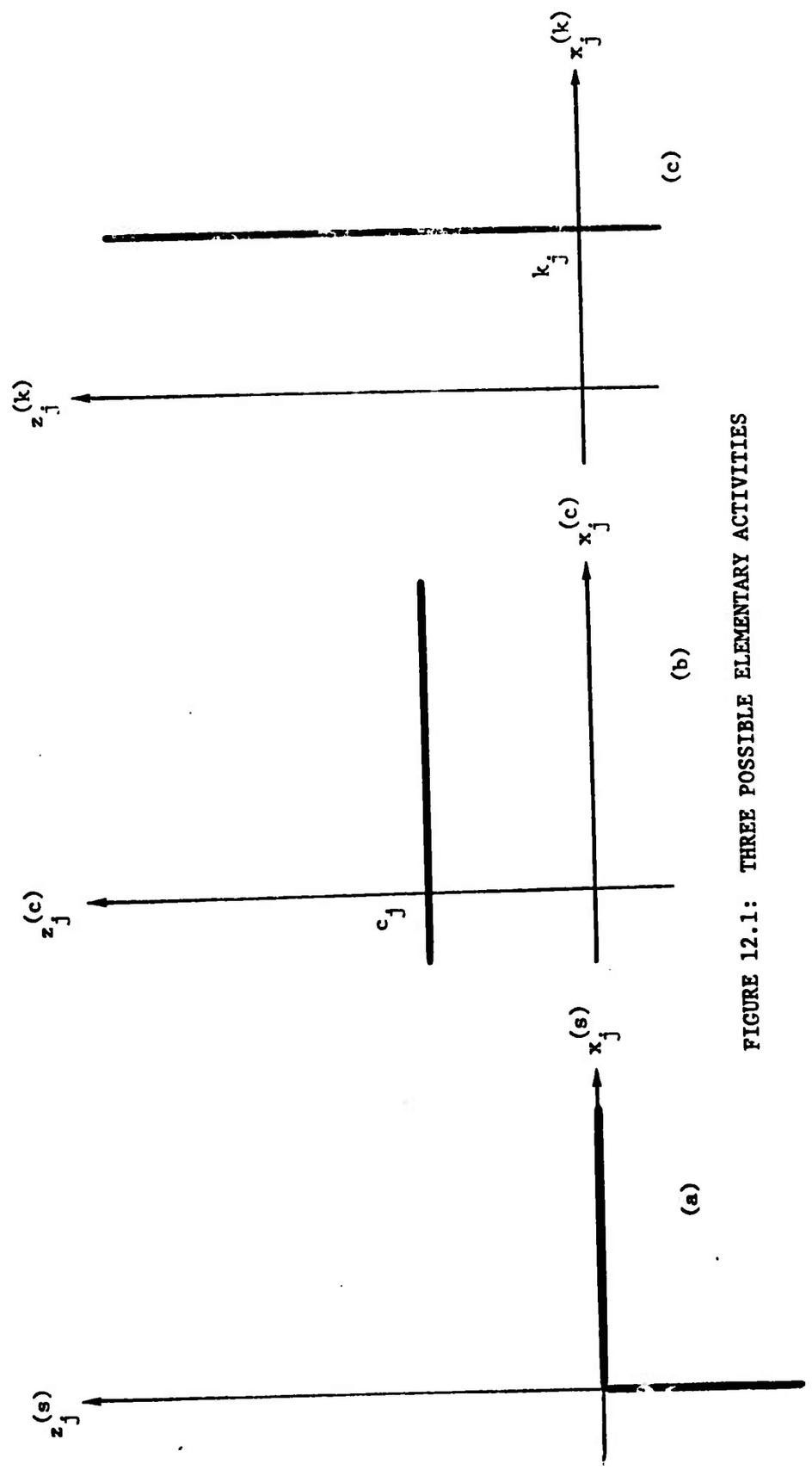


FIGURE 12.1: THREE POSSIBLE ELEMENTARY ACTIVITIES

To build up composite activities, we use the following construction rules between two arbitrary elements with given optimality diagrams, $(x_1;z_1)$ and $(x_2;z_2)$, to construct a new activity $(x_o;z_o)$.

(a) *"Series" Combination*

$$x_o = x_1 = x_2 \; ; \; z_o = z_1 + z_2$$

(b) *"Parallel" Combination*

(12.2)

$$x_o = x_1 + x_2 \; ; \; z_o = z_1 = z_2$$

plus the unary operation:

(c) *Transformation*

$$x_o = Kx_1 \; ; \; z_o = Kz_1 \; .$$

In this way, compound activities with very general "staircase" diagrams can be built up. For example, the compound activity shown in Figure 12.2 (a) can be built up out of the five elementary activities shown in Figures 12.2 (b) - (f) by adding the three constraints

$$\begin{aligned} x_o &= + x_2 + x_5 \\ 0 &= - x_2 + x_3 \\ 0 &= -x_1 - x_3 + x_4 \end{aligned} \qquad (12.3)$$

and substituting x_o in whatever other constraints hold for the compound activity. Rearrangements of (12.3) suggest the various ways in which the "assembly" of $(x_o;z_o)$ can occur.

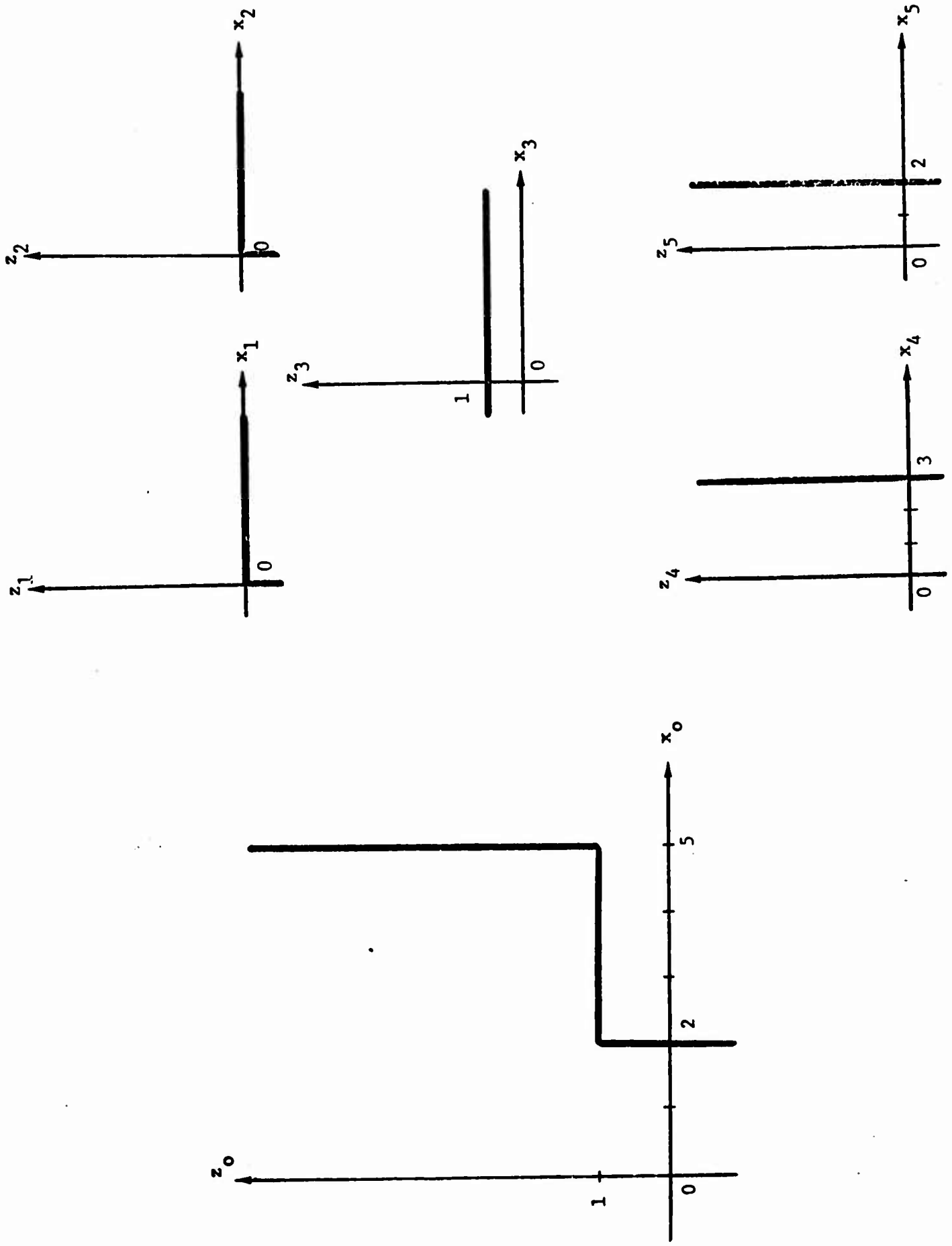


FIGURE 12.2: SHOWING HOW ELEMENTARY ACTIVITIES ARE USED IN CONSTRUCTING A COMPOUND ACTIVITY

Such an expanded representation naturally leads to many more elementary activities than real ones. But, conversely, the various simplex operations degenerate considerably for the types (12.1), and it may be possible to devise special algorithms for hand calculations, particularly when the problem is "weakly connected".

The use of elementary activities is nothing more than a faithful transcription of the *idealized current elements* (diode, voltage source, current source) in J. B. Dennis' 1958 thesis [8]. The notion of breakpoint stepping and operating with the optimality diagrams are also his; and, from a certain viewpoint, this entire paper might be said to be a natural interpretation of his method.

But, for some reason which this author does not understand, very little attention has been paid to this fundamental paper; in the next section we shall see that Dennis' breakpoint-stepping ideas can also be used in a larger context.

13. EXTENSIONS

Wolfe [16] has shown how quadratic programs can be calculated by applying complementary slackness conditions to an enlarged problem in which the primal and dual variables are "coupled" by linear relationships. The basic outline of an out-of-kilter method for these problems was given by Dennis [8], who introduced the idealized elementary activity which has a straight line $y_k = R_k x_k$ optimality diagram (a "resistor"); there appears to be no difficulty in extending this approach in the directions we have outlined here. Recently, Lemke [12] has shown that bimatrix games can also be put in complementary pivot form, which would then represent another extension for arbitrary starting solutions. In both these cases, the breakpoint curve consists of straight lines at arbitrary angles, as well as horizontal and vertical segments, and the appropriate theory may be found in [8], Chapters 5 and 6.

In principle, it would also be possible to extend these ideas to general convex programming, provided that one knew how to move both primal and dual solutions simultaneously along the monotone optimality curves defined by the Legendre transformation which gives the dual problem. For this reason, nonlinear solution methods appear to depend more strongly on special initial conditions than linear methods do. Another well-exploited procedure is to use quadratic programming as a local approximation (Chapter 7 of [8]).

REFERENCES

[1] Balinski, M. L., and R. E. Gomory, "A Primal Method for Assignment and Transportation Problem," Management Science, Vol. 10, No. 3, pp. 578-593, (April 1964).

[2] Balinski, M. L., and R. E. Gomory, "A Mutual Primal-Dual Simplex Method," Recent Advances in Mathematical Programming, R. L. Graves and P. Wolfe (eds.), McGraw-Hill, New York, pp. 17-26, (1963).

[3] Cottle, R. W., and G. B. Dantzig, "Complementary Pivot Theory of Mathematical Programming," Technical Report No. 16, Department of Operations Research, Stanford University, (June 1967).

[4] Charnes, A., and W. W. Cooper, MANAGEMENT MODELS AND INDUSTRIAL APPLICATIONS OF LINEAR PROGRAMMING, Volume I, John Wiley & Sons, New York, (1961).

[5] Dantzig, G. B., LINEAR PROGRAMMING AND EXTENSIONS, Princeton University Press, Princeton, (1963).

[6] Dantzig, G. B., and R. W. Cottle, "Positive (Semi-) Definite Matrices and Mathematical Programming," ORC 63-18, Operations Research Center, University of California, Berkeley, (May 1963).

[7] Dantzig, G. B., L. R. Ford, and D. R. Fulkerson, "A Primal-Dual Algorithm," in H. W. Kuhn and A. W. Tucker (eds.), "Linear Inequalities and Related Systems," Annals of Mathematics Study, No. 3, Princeton University Press, Princeton, pp. 171-181, (1956).

[8] Dennis, J. B., MATHEMATICAL PROGRAMMING AND ELECTRICAL NETWORKS, John Wiley & Sons, New York, (1959).

[9] Ford, L. R., Jr., and D. R. Fulkerson, FLOWS IN NETWORKS, Princeton University Press, Princeton, (1962).

[10] Jewell, W. S., "Optimal Flow Through Networks," Interim Technical Report No. 8, (Sc.D. Thesis), Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts, (June 1958). (Out of print.) The two main algorithms of this report were republished as:
   a. "Optimal Flow with Gains," Operations Research, Vol. 10, No. 4, pp. 476-422, (July-August 1962).
   b. "A Primal-Dual Multicommodity Flow Algorithm," ORC 66-24, Operations Research Center, University of California, Berkeley, (September 1966).

[11] Kelley, J. E., "Parametric Programming and the Primal-Dual Algorithm," Operations Research, Vol. 7, No. 3, pp. 327-334, (May-June 1959).

[12] Lemke, C. E., "Bi Matrix Equilibrium Points and Mathematical Programming," Management Science, Vol. 11, No. 7, pp. 681-689, (May 1965).

[13] Orchard-Hays, W., MATRICES, ELIMINATION AND THE SIMPLEX METHOD, C-E-I-R, Inc., Arlington, (1961).

[14] Simonnard, M., LINEAR PROGRAMMING, Prentice-Hall, New York, (1966).

[15] Wolfe, P., "The Composite Simplex Algorithm," SIAM Review, Vol. 7, No. 1, pp. 42-54, (January 1965).

[16] Wolfe, P., "The Simplex Method for Quadratic Programming," Econometrica, Vol. 27, No. 3, (July 1959).

APPENDIX A. ORGANIZING THE INCREMENTAL PROGRAM TABLEAU

Because the incremental program subroutine (2.2) and (2.3) is in a special form, in this section we consider how calculations *might* be carried out in tableau format. Naturally, to develop an efficient procedure for computer calculations, many other factors, such as size of storage, round-off error, matrix sparseness, etc. must be considered. As discussed in Section 8E, special algorithms may also be available for problems with special structure. However, the procedure outlined below is adequate for hand calculations and as a starting point for planning computer organization.

There are two novel features of the incremental program:

(a) it is homogeneous;

and (b) its variables are doubly bounded.

The first fact gives an easy starting solution in all cases; there is also no right-hand side to keep track of. To use (b) efficiently, the upper and lower bounding relationships are calculated outside the main tableau, and we keep separate track of the current values of the "nonbasic" variables which are at their upper or lower bounds.

A suggested format, as it might appear at the start of the first iteration is shown in Figure A1. The coefficients (a_{ij}) are placed in the main $m \times n$ array. In the next row above are placed the "pricing" terms, $\zeta_j + \varepsilon_j$, with

$$\text{(A1)} \qquad \varepsilon_s = \begin{cases} +1 & \text{if } \xi_s \text{ is to be maximized} \\ -1 & \text{if } \xi_s \text{ is to be minimized} \end{cases}$$

$$\varepsilon_j = 0 \qquad j \neq s^{\dagger}$$

---

† But see Section 8A for other possibilities.

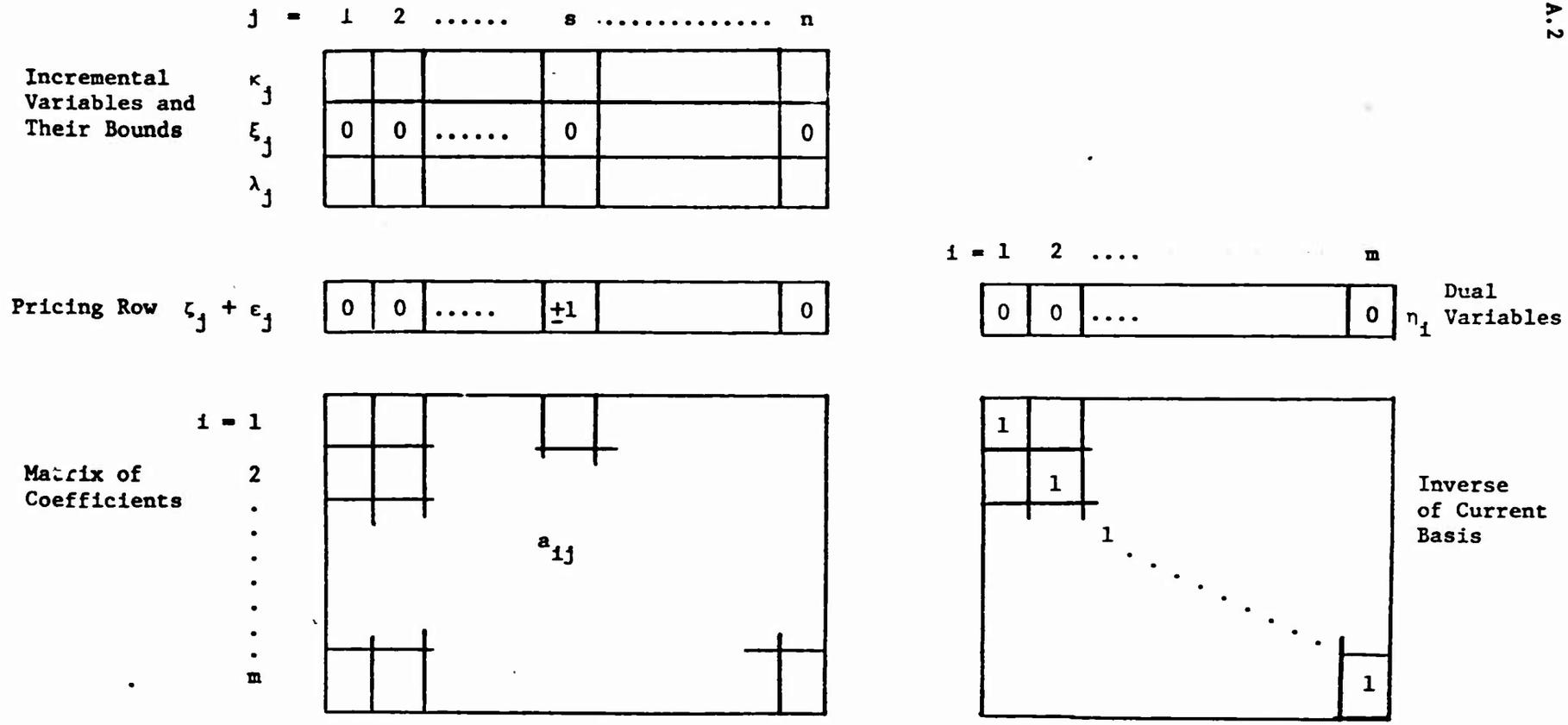


FIGURE A1: ORGANIZATION OF TABLEAU, SHOWING INITIAL VALUES AT THE BEGINNING OF THE FIRST ITERATION

Since the current values of the variables are "lost" during an upper- (and lower-) bounding technique, the $\{\xi_j\}$ are displayed, together with the current values of the $\{\kappa_j\}$ and $\{\lambda_j\}$, in three rows above the pricing terms.

For completeness, an $m \times m$ unit matrix is adjoined to the matrix of coefficients so that the dual variables $\{\eta_i\}$ and the inverse of the current basic are obtained automatically. In a computer code, a more efficient scheme such as the product form of the inverse would be used.

Once the initial basic set of variables, $\mathcal{J} = \mathcal{J}^0$, is specified, then the tableau can be reduced to the normal form shown in Figure A2 by the usual Gauss-Jordan reduction procedure, and the relative prices $\{\zeta_j + \varepsilon_j\}$ reduced to zero for $j \in \mathcal{J}$. This simultaneously gives the inverse of the current basis and dual variables for the incremental program. Pivoting operations are carried out in each application of the subroutine using the rules below; the last form of the tableau in a given iteration can be used to begin the next application of the subroutine (Appendix B).

In a doubly-bounded procedure, all basic variables satisfy:

$$\lambda_j \leq \xi_j \leq \kappa_j \quad ; \quad \zeta_j + \varepsilon_j = 0 \quad j \in \mathcal{J}. \tag{A2}$$

The nonbasic variables are at their upper or lower bound, and, *at optimality* (2.9):

$$\begin{aligned} &\text{If } \zeta_j^* + \varepsilon_j > 0 \qquad \xi_j^* = \kappa_j \\ &\text{If } \zeta_j^* + \varepsilon_j < 0 \qquad \xi_j^* = \lambda_j \end{aligned} \quad j \notin \mathcal{J}. \tag{A3}$$

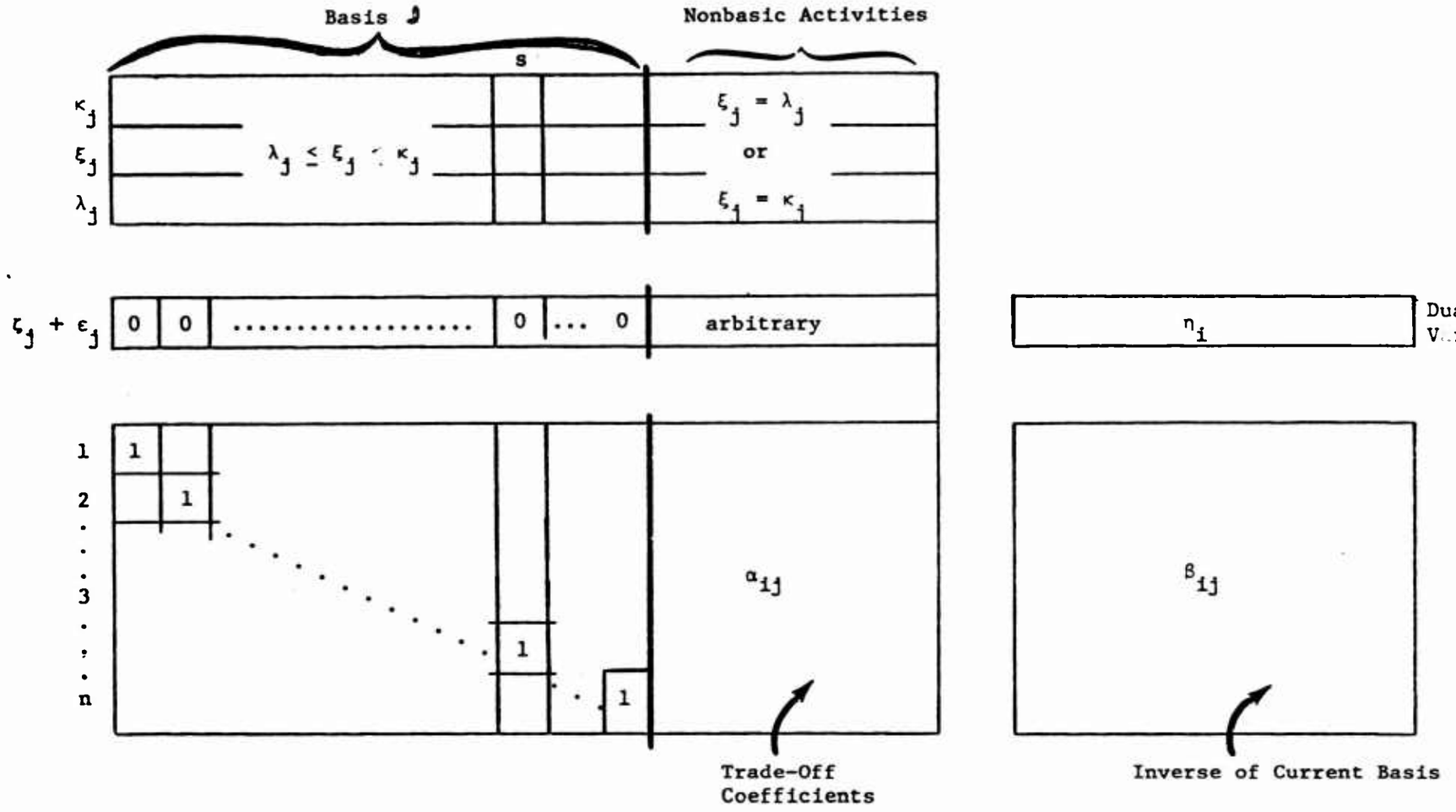


FIGURE A2: TABLEAU, SHOWING BASIC SET OF VARIABLES AND VALUES DURING ARBITRARY ITERATION

Thus,

Rule for Selection of a Candidate to Enter the Basis

Scan the Pricing Row for Some Nonbasic Activity $e \notin \mathcal{J}$ *such that*

(A4)
$$\text{(a)}\quad \zeta_e + \varepsilon_e > 0 \quad \text{and} \quad \xi_e < \kappa_e$$
$$\text{or}$$
$$\text{(b)}\quad \zeta_e + \varepsilon_e < 0 \quad \text{and} \quad \xi_e > \lambda_e \ .$$

(i) *If none exist, the current* $\{\xi_j\}$, $\{\eta_i\}$ *and* $\{\zeta_j\}$ *are optimal*

(ii) Otherwise, ξ_e is *a candidate to enter the basis.*

Now, if some ξ_e is to enter the basis from its lower bound, a positive [negative] *trade-off coefficient,* α_{ie} , in row i , column e of the tableau indicates that the basic variable $\xi_{\ell(i)}$ corresponding to row i (i.e., the unit matrix has an entry in column ℓ , row i) must decrease [increase]: the bound $\lambda_{\ell(i)}[\kappa_{\ell(i)}]$ may thus limit the increase of ξ_e , as will its own bound, κ_e . Similar remarks apply if ξ_e is to enter the basis from its upper bound.

The smallest such change determines the actual change to be made in ξ_e : if it is a basic variable $\xi_{\ell(i)}$ which reaches a bound, ξ_e replaces $\xi_{\ell(i)}$ in the basis, *with* α_{ie} *as pivot; if* ξ_e goes all the way to its other bound, the *basic set of variables remains unchanged,* and only the values of the ξ_j are changed. Obvious remarks apply to ties and unboundedness.

Thus follows:

Rule for Selection of Candidate to Leave the Basis, and for Changing the Values of the Incremental Variables

(A5) 1. For a given candidate activity e , a nonzero trade-off coefficient α_{ie} , and the corresponding basic activity $\ell(i)$, partition $\mathcal{J}$ into the following sets, and define the indicated *displacements,* δ_j , for all $j \in \mathcal{J} + \{e\}$

(A6)

(a) $\ell(i) \in \mathcal{J}^+$ iff $\begin{cases} \zeta_e + \varepsilon_e > 0 \text{ and } \alpha_{ie} < 0 \\ \text{or} \\ \zeta_e + \varepsilon_e < 0 \text{ and } \alpha_{ie} > 0 \end{cases}$; $\delta_\ell = \begin{cases} (\kappa_\ell - \xi_\ell)/(-\alpha_{ie}) \\ (\kappa_\ell - \xi_\ell)/(+\alpha_{ie}) \end{cases}$

(b) $\ell(i) \in \mathcal{J}^-$ iff $\begin{cases} \zeta_e + \varepsilon_e > 0 \text{ and } \alpha_{ie} > 0 \\ \text{or} \\ \zeta_e + \varepsilon_e < 0 \text{ and } \alpha_{ie} < 0 \end{cases}$; $\delta_\ell = \begin{cases} (\xi_\ell - \lambda_\ell)/(+\alpha_{ie}) \\ (\xi_\ell - \lambda_\ell)/(-\alpha_{ie}) \end{cases}$

(c) Otherwise $\ell(i) \in \mathcal{J} - \mathcal{J}^+ - \mathcal{J}^-$; $\delta_\ell = \infty$

(d) Define $\delta_e = \begin{cases} \kappa_e - \xi_e & \text{if } \zeta_e + \varepsilon_e > 0 \\ \xi_e - \lambda_e & \text{if } \zeta_e + \varepsilon_e < 0 \end{cases}$.

2. Compute the *allowed displacement*, δ .

(A7)
$$\delta = \min \left\{ \delta_e ; \min_{\ell \in \mathcal{J}^+} \delta_\ell ; \min_{\ell \in \mathcal{J}^-} \delta_\ell \right\} .$$

(a) If $\delta = \infty$, the incremental program is unbounded. (Return to main program with $\Delta^* = \infty$.)

(b) If the minimum δ in (A7) is computed from $\mathcal{J}^+$ or $\mathcal{J}^-$, then activity e enters the basis at level $\delta < \delta_e$, driving some basic activity ℓ to its lower or upper bound, whence it leaves the basis. Pivot on the appropriate α_{ie} and reduce all rows of the tableau plus the pricing row to normal form with the new basis $\mathcal{J}' = \mathcal{J} + \{e\} - \{\ell\}$.

(c) If δ is determined by δ_e , then ξ_e goes to a bound, and the current basis $\mathcal{J}$ remains unchanged. $\mathcal{J}' = \mathcal{J}$.

(d) In case of a tie, the choice is immaterial (avoid a pivot operation, if possible).

3. Compute the new values of the $\{\xi_j\}$:

$$\xi_e = \xi_e \begin{Bmatrix} + \delta \\ - \delta \end{Bmatrix} \text{ if } \begin{Bmatrix} \zeta_e + \varepsilon_e > 0 \\ \zeta_e + \varepsilon_e < 0 \end{Bmatrix}$$

$$\begin{aligned} \xi_j &= \xi_j + \delta \quad \text{if} \quad j \in \mathcal{J}^+ \\ &= \xi_j - \delta \quad \text{if} \quad j \in \mathcal{J}^- \\ &= \xi_j \qquad \text{if} \quad j \in \mathcal{J} - \mathcal{J}^+ - \mathcal{J}^- \ . \end{aligned}$$

Then, a new candidate is selected, using (A4) , and the steps are repeated until the optimality criterion (A3) is satisfied. Control is then returned to the main routine, with the optimal values $\{\xi_j^*\}$, $\{\eta_i^*\}$, and $\{\zeta_j^*\}$ from the appropriate points of the tableau, and $\Delta^* = \sum \varepsilon_j \xi_j^*$.

The next call for the subroutine will clear $\{\xi_j\}$, ..., and give new values to $\{\lambda_j\}$ $\{\kappa_j\}$ and $\{\varepsilon_j\}$; however, the rest of the tableau can be saved in its last (reduced) form, with the optimal basis $\mathcal{J}^*$ from this iteration (Appendix B).

APPENDIX B. SELECTING THE INITIAL AND SUBSEQUENT BASIS

To get the incremental program subroutine started, it is necessary to specify an initial basic set $\mathcal{J}^0$. However, since $\xi_j = 0$ for *all* variables at the beginning of each of the subroutines, *any* selection of m-independent columns of (a_{ij}) will work. Typically, some slack or error variables were added in the formulation which provide convenient unit vectors. At worst, with a full matrix, an arbitrary element can be selected to clear its column; any dependent columns will show up as columns of zeroes and cannot be used. Note that *"frozen" variables*

$$0 \leq \xi_j \leq 0 \qquad (\lambda_j = \kappa_j = 0)$$

can conveniently be regarded either as basic variables, or as nonbasic variables at either bound.

Once the tableau has been put in normal form, it is probably most efficient and convenient to leave it in that form, making further changes only by pivot operations. To do this, the following result is needed:

(B1) *The optimal basic set, $\mathcal{J}^{t*}$, from one iteration of the program subroutine may be used as the starting basic set $\mathcal{J}^{t+1}$ for the next iteration.*

The proof is trivial, once we note (2.9), the fact that all ξ_j are reset to zero for the next iteration, and the remark about frozen variables made above.

Suppose the objective coefficients at iteration t were ε_j^t $(j = 1, \ldots, n)$; the optimal values of the $\{\xi_j^*\}$ can be determined directly from the pricing row of the first tableau by subtracting $\{\varepsilon_j^t\}$. Then, if the same basic set, $\mathcal{J}^{t*}$, is kept to start the $(t + 1)^{st}$ iteration, one changes the nontableau rows

to the new values as follows:

$$\xi_j^{t+1} = 0 \ ;$$

$$\kappa_j^{t+1} \ , \ \lambda_j^{t+1} \ , \text{ following (3.2)}$$

(B2)

$$(\text{pricing row})^{t+1} = (\text{pricing row})^{t} - \{\varepsilon_j^t\} + \{\varepsilon_{j+1}^t\}$$

$$\eta_i^{t+1} = \eta_i^* \text{ (at iteration } t \text{)}$$

with the tableau and basis inverse remaining unchanged.

Note that if one is still working on the same activity (or group of activities) in the next iteration, only the bounds will change as a result of the primal or dual displacements. Since the number of such changes is usually small (one or two) one could take

$$\kappa_j^{t+1} = \kappa_j^t - \xi_j^*$$

(B3)

$$\lambda_j^{t+1} = \lambda_j^t + \xi_j^*$$

first, and then modify the bounds only for the activities which changed state in the dual step (6.4).

APPENDIX C. THE INCREMENTAL PROGRAM TABLEAU FOR THE SYMMETRIC PROBLEM

In the symmetric problem of Section 9, a negative unit matrix is in the formulation explicitly; by extending the index of ξ_k from 1 to $m + n$, we get the incremental changes on the $\{r_i\}$ as well as the $\{x_j\}$. Figure C1 shows how the tableau might be organized; $\mathcal{J}^o$ might as well be taken as the last m columns. Note that bounds and pricing now apply to all $m + n$ columns; if one of the last m columns is basic, a slack change may occur. At optimality, the pricing row actually contains $\zeta_k^* + \epsilon_k$, $(k = 1,2, \ldots, n)$ or $-\eta_{k-n}^* + \epsilon_k$, $(k = n+1, n+2, \ldots, n+m)$.

If there is a "slack" column in $\mathcal{J}$ which prices out incorrectly at some iteration, we see that clearing the pricing row to zero in that column will introduce price changes in the first n columns, and thus a nonslack variable will become a candidate to enter $\mathcal{J}$. This step will in fact be a *dual-pivot* as described in the dual simplex algorithm, Appendix D.D. A more compact, symmetric form could be obtained from a *condensed tableau* form of C1 with rather complicated row and column-pivot rules [2]; however, rather than strive for complete symmetry, we leave the procedure here in its expanded form, so that the (negative) inverse of the current basis is always available.

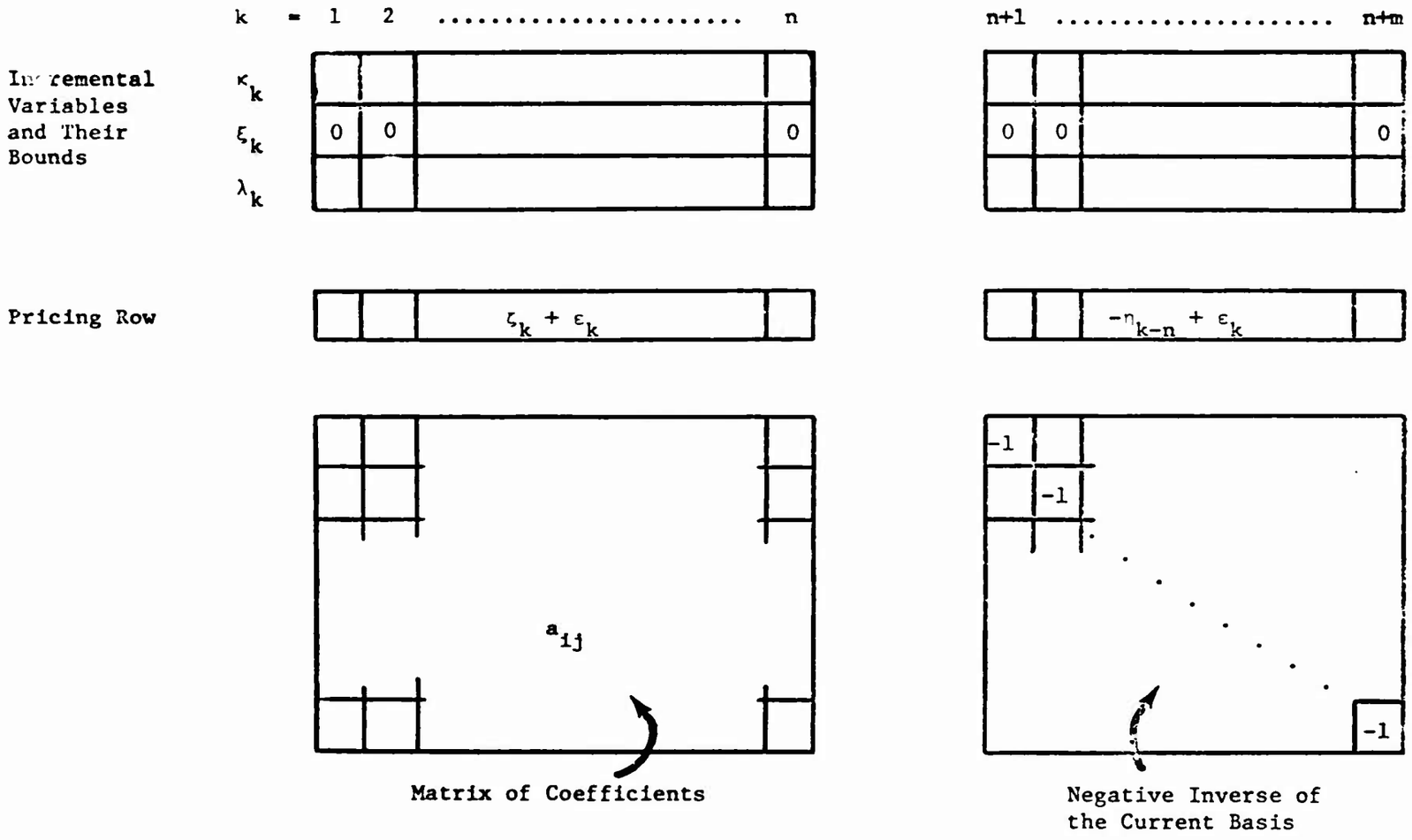


FIGURE C1: INITIAL TABLEAU FOR SYMMETRIC PROBLEM

APPENDIX D. COMPARISON WITH OTHER SIMPLEX ALGORITHMS

To fully appreciate the relationship between the COMPLEX method and the many special algorithms which are in current use, it is instructive to explain the possible variants in terms of the optimality diagram.

A. The Primal Simplex Algorithm

In its primitive form, a problem is formulated using unbounded, nonnegative variables; nonnegative "slack" variables are added at no cost as needed to change all constraints to equalities. It is assumed that a *primal-feasible basis,* B^p of nonnegative $\{x_j\}$ is known; the nonbasic variables have value $x_j = 0$, $j \notin B^p$.

Dual variables are determined by solving:

(D1) $$z_j = \sum y_i a_{ij} = c_j \qquad j \in B^p$$

and thus we may have:

(D2) $$z_j \begin{cases} < c_j \\ = c_j \\ > c_j \end{cases} \qquad j \notin B^p$$

(If $z_j = c_j$, we may perturb the c_j, so that B^p corresponds only to our set B).

On the simplified complementary-slackness diagram (Figure D1), all basic variables ① are on B, while nonbasic variables are either conforming in L(②), or *nonconforming* ③ (in K^-) *with* $x_j = 0$.

An arbitrary activity, e, of type ③ is selected to enter the basis, and $x_e = \xi_e$ is maximized while primal-freezing all other nonbasic variables, *including the other nonconforming ones;* horizontal movement of ξ_e is unlimited until some activity ℓ in B reaches its lower bound zero, and thus leaves the basis and is considered as an activity of type ②. As discussed in Section 8B, this means that the incremental subroutine consists of a single pivot operation; the dual changing step determines θ so as to make activity e dual-conforming,

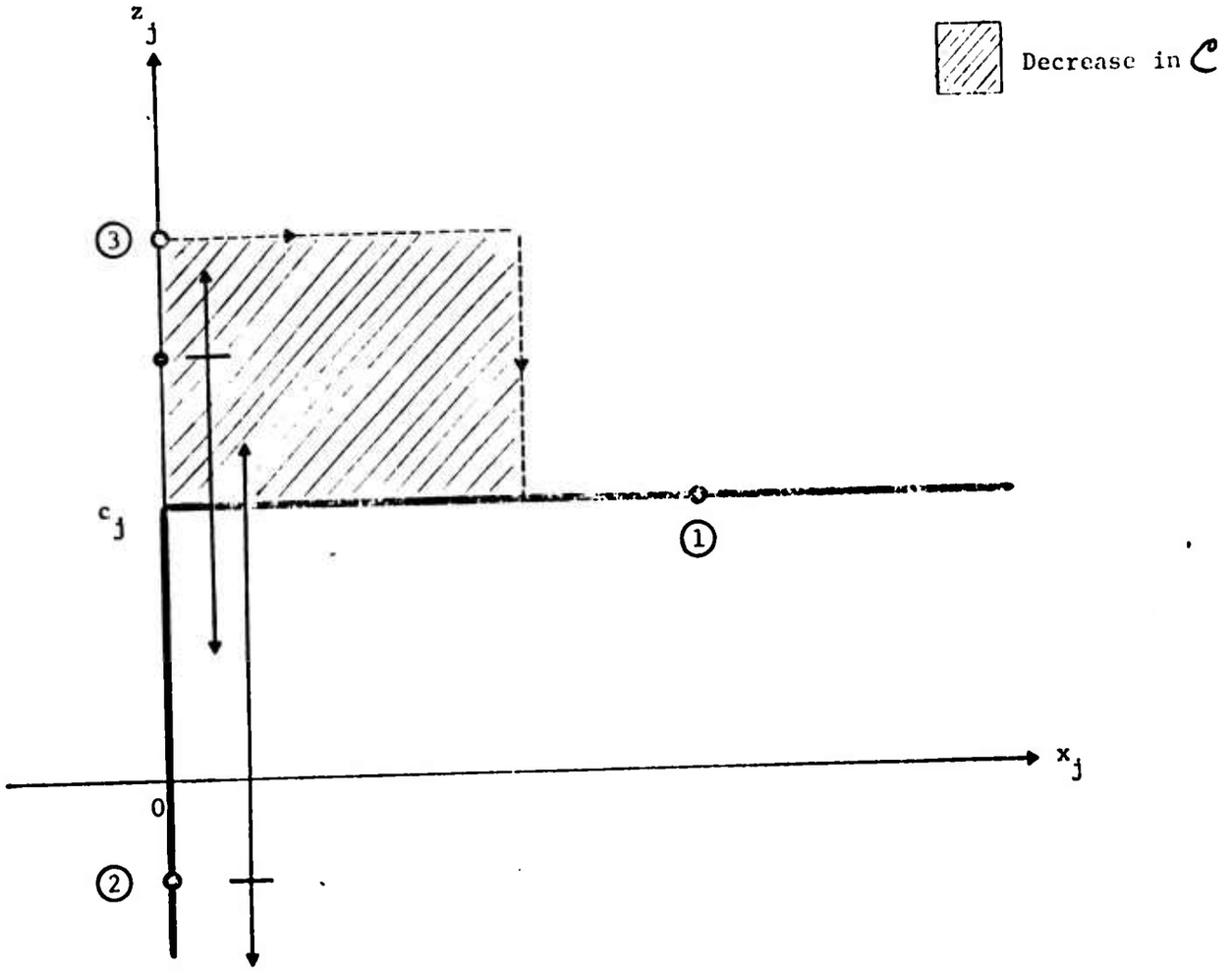


FIGURE D1: OPTIMALITY DIAGRAM FOR SIMPLEX ALGORITHM SHOWING BASIC AND NONBASIC ACTIVITIES, AND PIVOT STEP

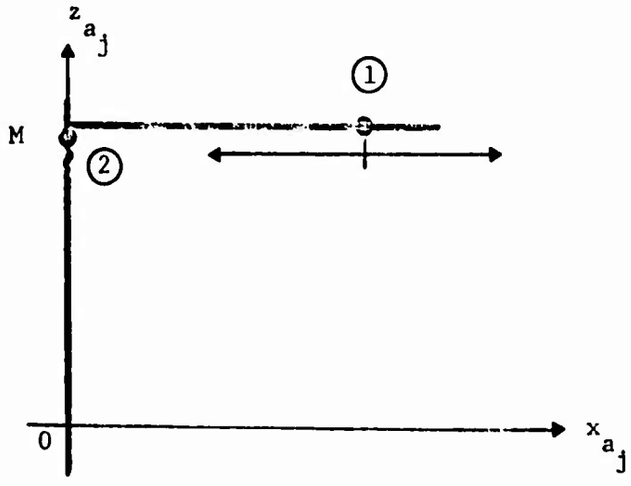
since it is "brought into the basis B^p " (i.e., will be used to calculate the z_j from (D1)). This forced conformity may possibly lead to dual "overshooting," some conforming nonbasic activities become nonconforming, and vice versa. (Activity ℓ , however, will always move into L .)

The single-pivot procedure is repeated with the new (possibly enlarged) set of nonconforming activities, until all are made conforming, basic or not. At each step, the decrease in the primal functional $\mathcal{C}$ is the shaded area shown, and convergence (except for degeneracy) argued on this fact.

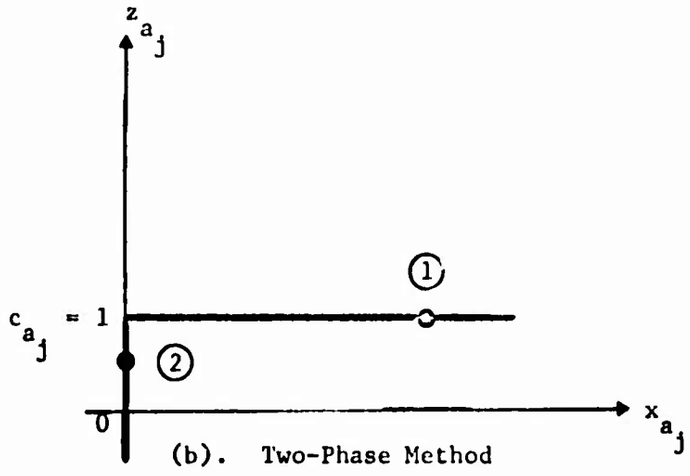
Sometimes the candidate e is not selected arbitrarily, but is chosen from among those which are "highest above B ", or "will give the greatest area" on their respective diagrams.

B. Artificial Variables

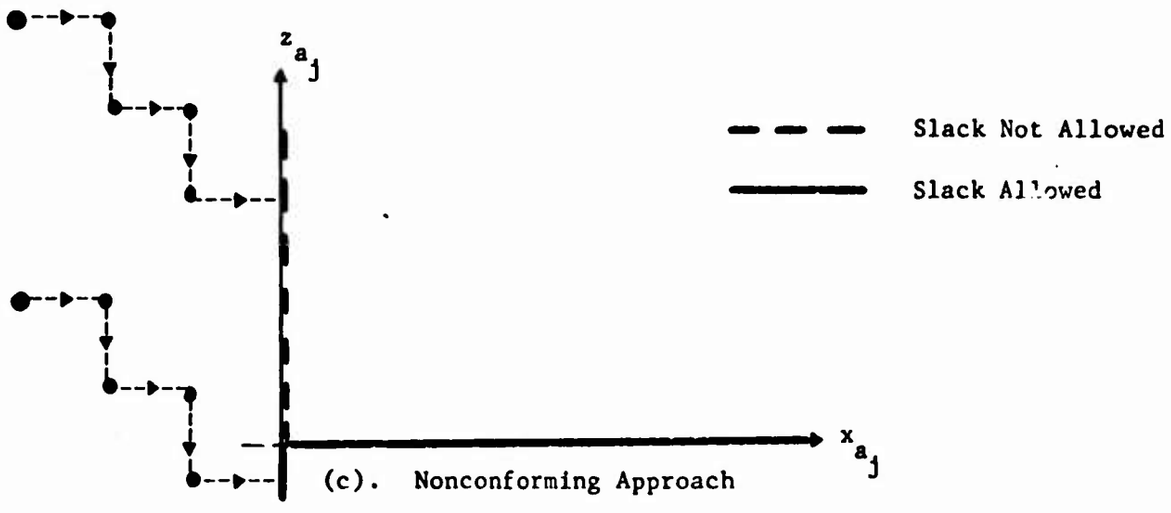
Getting a primal-feasible basis B^p can be a nuisance. If a unit matrix of appropriate sign cannot be found in (a_{ij}) , then *nonnegative artificial variables*, x_{a_j} , are placed in B^p as needed. Since they must ultimately satisfy $x_{a_j} = 0$, they are given a transfinite cost M (Figure D2a) , but are considered conforming. In this "method of penalties", solution of (D1) gives transfinite "nonconformities" $M(z_j - c_j)$ to some of the ordinary variables; then application of the primal simplex algorithm removes these artificial variables, one at a time, to position ② in Figure D2a whence they are ignored. (Assuming primal feasibility). Since finite $z_j - c_j$ discrepancies are ignored during this procedure, further finite steps are still needed to correct any activities which are or become finite nonconforming.



(a). Method of Penalties



(b). Two-Phase Method



(c). Nonconforming Approach

FIGURE D2: OPTIMALITY DIAGRAMS FOR ARTIFICIAL VARIABLES

In the "two-phase method," transfinite numbers are avoided, but a separate cost functional, with:

$$c_j = \begin{cases} 1 & j \text{ artificial} \\ 0 & j \text{ ordinary} \end{cases} \tag{D3}$$

is used in "Phase 1" (Figure (D2b)). This procedure will (if the problem is primal-feasible) drive all the x_{a_j} to zero. In "Phase II", these variables are frozen at zero, and the ordinary functional C is taken up, since a primal-feasible basis B^p of ordinary activities is available.

As is well known, the above two methods are equivalent.

In the formulation of our algorithm, these variables are treated either as error variables which are to satisfy $x_{a_j} = 0$ (dotted line in Figure D2c), or as nonconforming (negative) values of slack variables (solid line in Figure D2c). In our approach, these would individually (or jointly) be made conforming, following the usual breakpoint procedure. To obtain the same sequence of steps as the above methods, however, one would need to:

(D4)
- (a) Take all of these artificial variables as (part of) the initial basic set $\mathcal{J}^0$;
- (b) Set $\varepsilon_j = 1$ for these variables in the subroutine (2.1);
- and (c) Ignore the nonconformity of the variables not in $\mathcal{J}$ until after all these special variables have been made conforming (i.e., until the next application of the subroutine).

C. Combined Methods

In various synthetic methods, it is proposed to combine Phase I and II by using

$$f(w) = \sum x_{a_j} + w \sum c_j x_j \tag{D5}$$

as a single objective function, with w as a parameter. Hopefully, for some finite w, this would eliminate artificial variables, and still give a solution not too far from the optimum. If not all $x_{a_j} = 0$, w would be decreased, and the minimization continued; in the worst case, it might be necessary to set $w = 0$, and enter a true Phase I to decide if the problem is infeasible.

This approach gives an obvious modification to subsection B above, or may be thought of as a cost-parametric procedure (Subsection G, below).

D. The Dual Simplex Algorithm

In the dual simplex algorithm, a *dual-feasible basis*, B^D, is given, such that when solving (D1), we have

$$\text{(D6)} \qquad \begin{aligned} z_j = \sum y_i a_{ij} = c_j \qquad & j \in B^D \\ z_j \leq c_j \qquad & j \notin B^D \end{aligned} .$$

The corresponding primal solution:

$$\text{(D7)} \qquad \begin{aligned} \sum_{j \in B^D} a_{ij} x_j = b_i \qquad & i = 1, \ldots, m \\ x_j = 0 \qquad & j \notin B^D \end{aligned}$$

must, then, have:

$$\text{(D8)} \qquad \text{Some } x_j < 0 \qquad j \in B^D$$

if optimal basis has not been picked. On the optimality diagram of Figure D3, then have dual-feasible solution points which are: ① basic, primal-feasible); ② nonbasic, zero-values (in L); or ③ basic, primal infeasible (in B^-).

An arbitrary activity, ℓ, in B^- is selected to *leave* the basis, (i.e., one maximizes ξ_ℓ), while dual-freezing all basic variables. The incremental subroutine and the dual-changing steps (6.4) (6.5) together constitute a single-pivot operation in which the basic variable ξ_ℓ is increased until it is conforming (i.e., $\kappa_\ell = x_\ell$): to select the vector to *enter* the basis, all activities in L *which move towards the right* are considered (i.e., all variables x_j such that $\alpha_{i(\ell),j} < 0$, where $i(\ell)$ is the row corresponding to basic activity ℓ). The actual vector e is chosen to minimize $(c_j - z_j)/ \mid \alpha_{i(\ell),\, j} \mid$, so that it would be the *first* to be conforming again, in state B, i.e., when it enters the basis at the dual-changing step (6.4) (6.5). (See also Subsection 8F.)

There is no difficulty with the other nonbasic variables, since they are not selected to enter the basis, and hence remain at level zero; however, if all $\alpha_{i(\ell),\, j}$ are ≥ 0 for some $i(\ell)$, then no candidate could enter the basis to replace ℓ, and the primal problem would be infeasible (an $\alpha_{i(\ell),\, e} = 0$ cannot be used as a pivot).

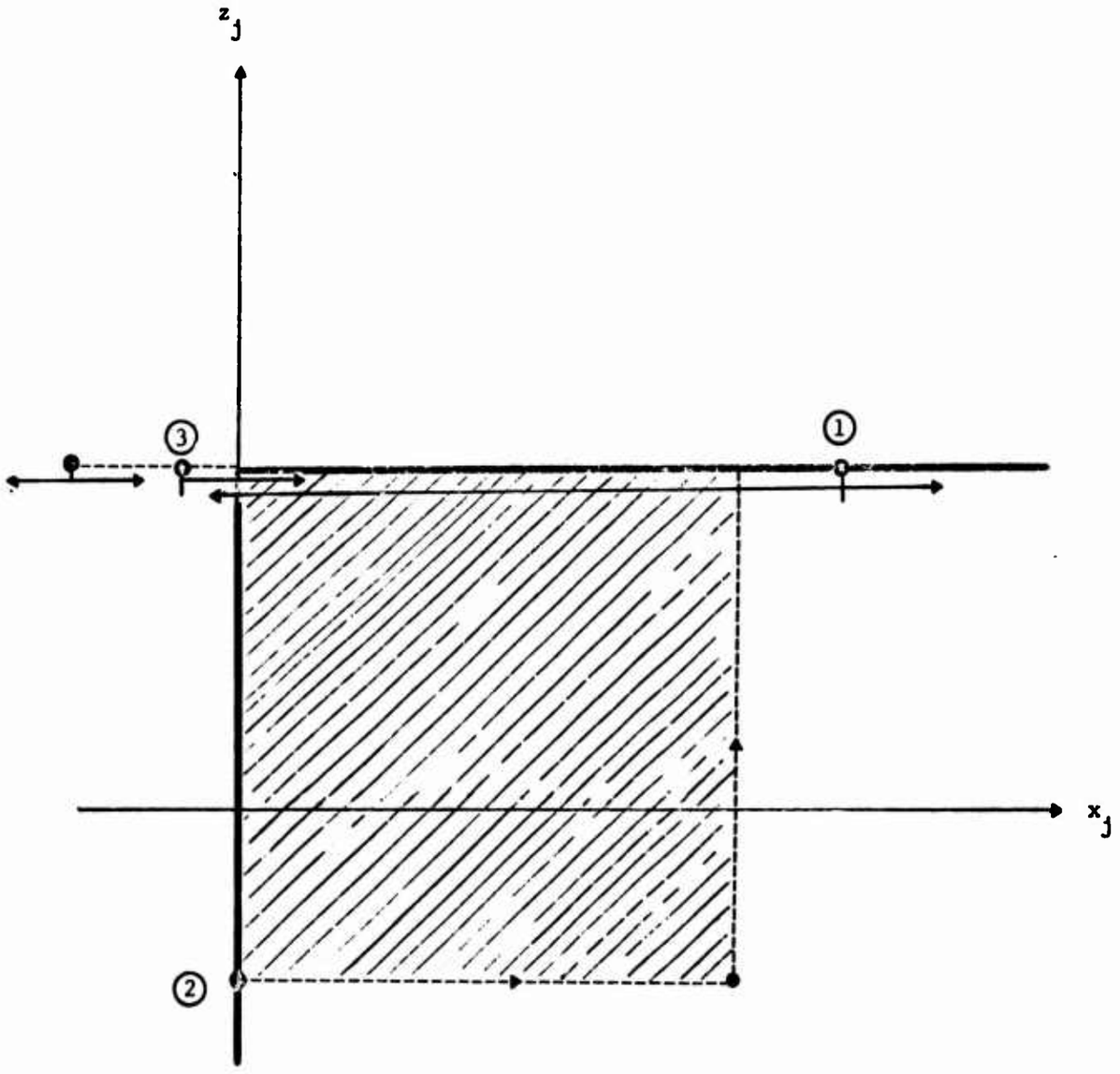


FIGURE D3: OPTIMALITY DIAGRAM FOR DUAL SIMPLEX ALGORITHM

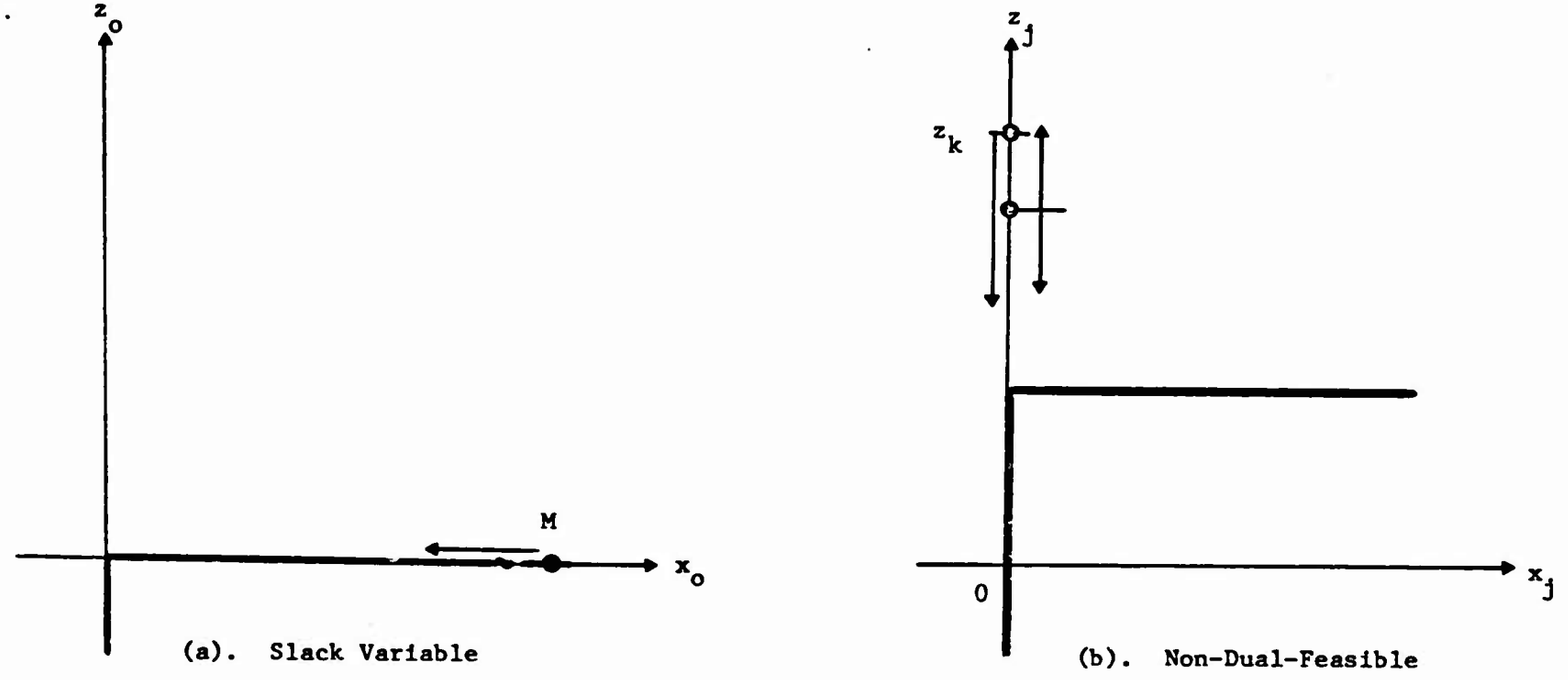
Forcing x_ℓ to leave the basis in one pivot may cause "overshooting" in the sense that some basic variable in B may become primal infeasible in B^-, and hence a future candidate to leave the basis; however, convergence is argued on the increase in the dual functional β which is equal to the shaded area in Figure D3. As in the primal simplex algorithm, special rules of choice, such as the "most nonconforming" (smallest $x_\ell < 0$), or "greatest shaded area" can be used.

E. The Artificial Constraint

The dual simplex method is usually applied *because* a dual-feasible basis is known, as in parametric problems. However, if a basis B^o is known which is not dual-feasible, then an artificial constraint

$$\sum_{j \notin B^o} x_j + x_o = M \tag{D9}$$

may be added, with x_o a slack activity. This gives an optimality diagram for x_o which looks like Figure (D4a) with an enlarged basis for the extended problem of $B^o + \{0\}$.



(a). Slack Variable

(b). Non-Dual-Feasible

FIGURE D4: OPTIMALITY DIAGRAMS FOR DUAL SIMPLEX ARTIFICIAL CONSTRAINTS

Now among the non-dual-feasible activities j, $j \notin B^o$, there is one which is "highest" above line B in Figure D4b, i.e., some k for which

$$z_k - c_k = \max_{\substack{j \in B^o \\ z_j > c_j}} [z_j - c_j] . \tag{D10}$$

By pivoting on x_k in (D9), all of these points are made conforming on L, with x_k replacing x_o in the now dual-feasible extended basis

$$B^D = B^o + \{k\} \tag{D11}$$

at level

$$x_k = M .$$

Unless x_o comes back into the basis at a positive level, the optimal solution may be unbounded.

F. The Primal-Dual Method

In the primal dual-method [7], one begins with a (not necessarily basic) dual feasible solution $\{y_i^c\}$, and a primal in which nonnegative artificial variables x_{a_j} have been added to every primal constraint. This gives the optimality diagrams shown in Figure D5a and b.

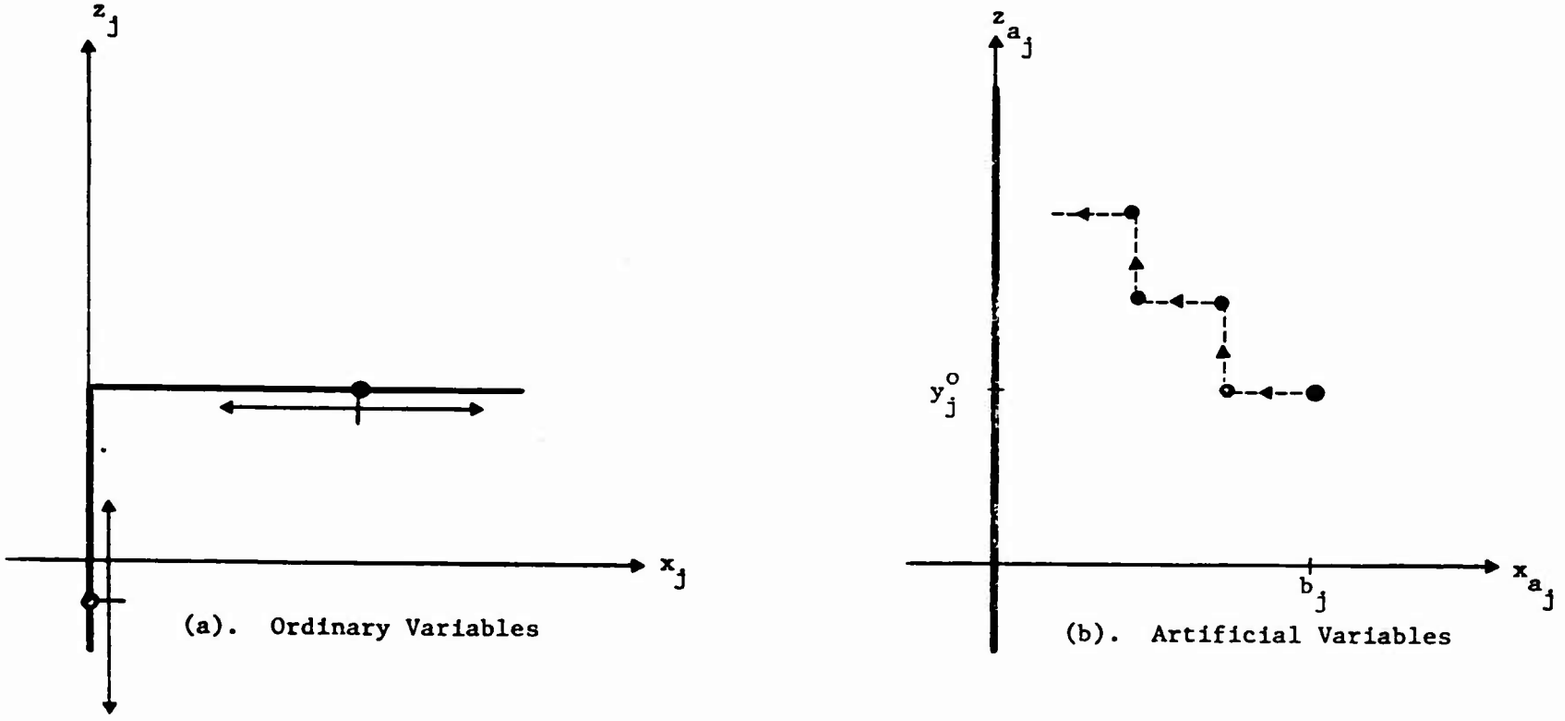


FIGURE D5: OPTIMALITY DIAGRAMS FOR THE PRIMAL-DUAL METHOD

The initial starting solutions are the nonbasic variables $x_j^o = 0$, and basic variables $x_{a_j}^o = b_j$ (note that $z_{a_j}^o = y_j^o$).

If $z_j^o < c_j$, then all such variables are "frozen" in L; only variables for which $z_j = c_j$ are allowed to vary in an *associated restricted primal*. The objective of this general linear programming problem is to minimize the error terms:

(D12) $$\text{Minimize } E = \sum x_{a_j},$$

while keeping all ordinary variables conforming (in L or B). When $E = 0$, the optimal solution is obtained.

This method can be directly explained as a special case of the algorithm of Section 6, when applied to nonnegative unbounded variables.

(a) The indices of the artificial variables are taken as the set J^o; these variables are nonconforming in L^+.

(b) The ordinary variables are considered to be conforming in L if $z_j - c_j < 0$, or in B if $z_j = c_j$.

(c) In applying the incremental subroutine, all nonconforming (artificial) variables are worked on simultaneously ($\varepsilon_j = -1$ for j nonconforming).

Otherwise, the primal-dual algorithm is identical to our algorithm with no special freezing or one-pivot rules being used (except possibly between artificial variables only). Successive applications of the incremental subroutine will reduce the artificial variables, following the usual breakpoint-stepping procedure, and the optimal solution to the restricted problem after one pass can be used as the starting solution for the next pass (the optimal J from one pass can be used to start the next one).

As is well known, in certain problems one may avoid putting in artificial

variables for all constraints; for example, in flow problems, the conservation equations are never violated, since each (incremental) maximal flow subroutine only adds loop flows. Also, for this class of problems the subroutine is solved by a particularly efficient labelling algorithm.

If no dual-feasible solution exists, then an artificial constraint

$$\sum_{j=1}^{n} x_j + x_o = M \tag{D13}$$

is added, and

$$\begin{aligned} y_o^o &= \min_j [0, c_j] \\ y_i^o &= 0 \qquad i = 1, \ldots, m \end{aligned} \tag{D14}$$

constitutes a dual-feasible solution to the augmented program.

G. Cost Parametrization

If a certain unit cost, c_j, is to be studied *parametrically*, starting from an optimal solution $(x_j^o ; z_j^o)$, then the substitution

$$c_j' = c_j + \theta \tag{D15}$$

is made with, say, increasing values of θ. The optimality diagram for this activity moves upward by an amount θ, leaving the current solution $(x_j^o ; z_j^o)$ either conforming in $L(x_j^o = 0)$, or nonconforming in $L^+(x_j^o > 0)$. (Figure D6)

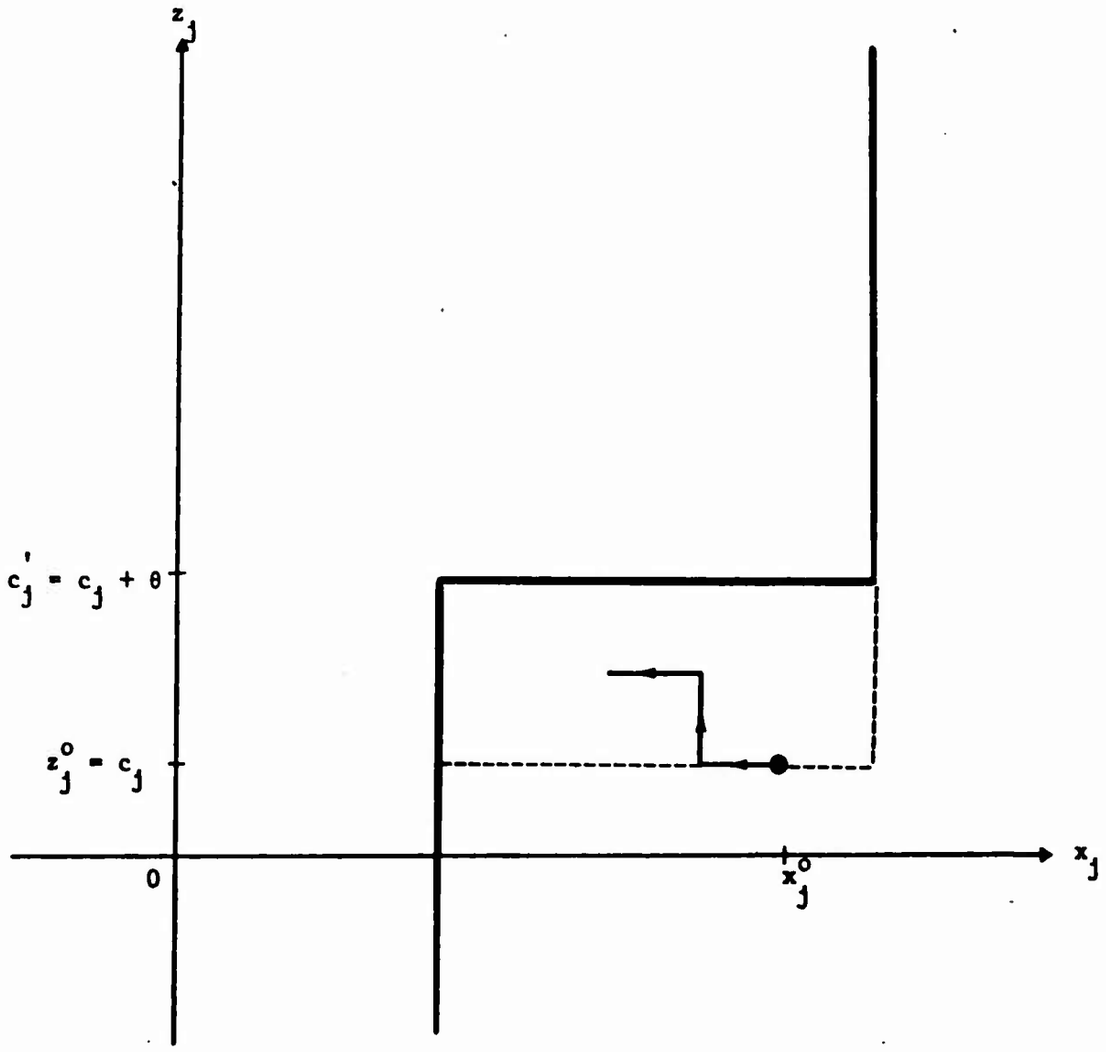


FIGURE D6: OPTIMALITY DIAGRAM FOR COST PARAMETRIZATION

In the first case, θ can increase without limit, activity j remaining nonbasic in L, with no change in $\mathcal{C}^o = \mathcal{D}^o$.

In the second case, j is the only nonconforming activity, so the subroutine seeks to minimize ξ_j, keeping all other activities conforming. After a possible finite decrease in x_j equal to $-\Delta_1^*$ at level c_j, a dual change is made, moving the current solution upwards by an amount θ_1^*, then to the left by an amount $-\Delta_2^*$, ..., etc., following the familiar breakpoint curve. Finally, after a finite number of steps, the solution will be at the desired level $z_j = c_j'$, or $x_j = 0$. Because the starting solution was optimal, except for j, only one basis change will usually be made in the subroutine as an activity in L changes to B or vice versa (except for degeneracy).

In the familiar tracing-out of the piecewise linear curve of $\mathcal{C}^*$ versus θ, the vertical displacements θ_1, θ_2, ..., etc., correspond to intervals between *turning points* of $\mathcal{C}^*$, and the determination of Δ^* corresponds to the basis change at these turning points. The decrease in $\mathcal{D}^*$ at the origin $\theta = 0$ is $-\Delta_1^* (c_j' - z_j^o)$, and the increase in $\mathcal{C}^*$ when θ changes to θ_1^* is $\theta_1^* (x_j^o + \Delta_1^*)$, etc.

Complementary remarks apply to a *decrease* in θ.

If a vector change in the c_j is proposed:

$$c_j' = c_j + \theta e_j \qquad j = 1,2, \ldots, n \tag{D16}$$

then the several nonconforming activities are worked upon at the same time by the incremental subroutine with $\varepsilon_j = -e_j$.

H. Right-Hand Side Parametrization

First, suppose some requirements value, b_i, is to be changed to:

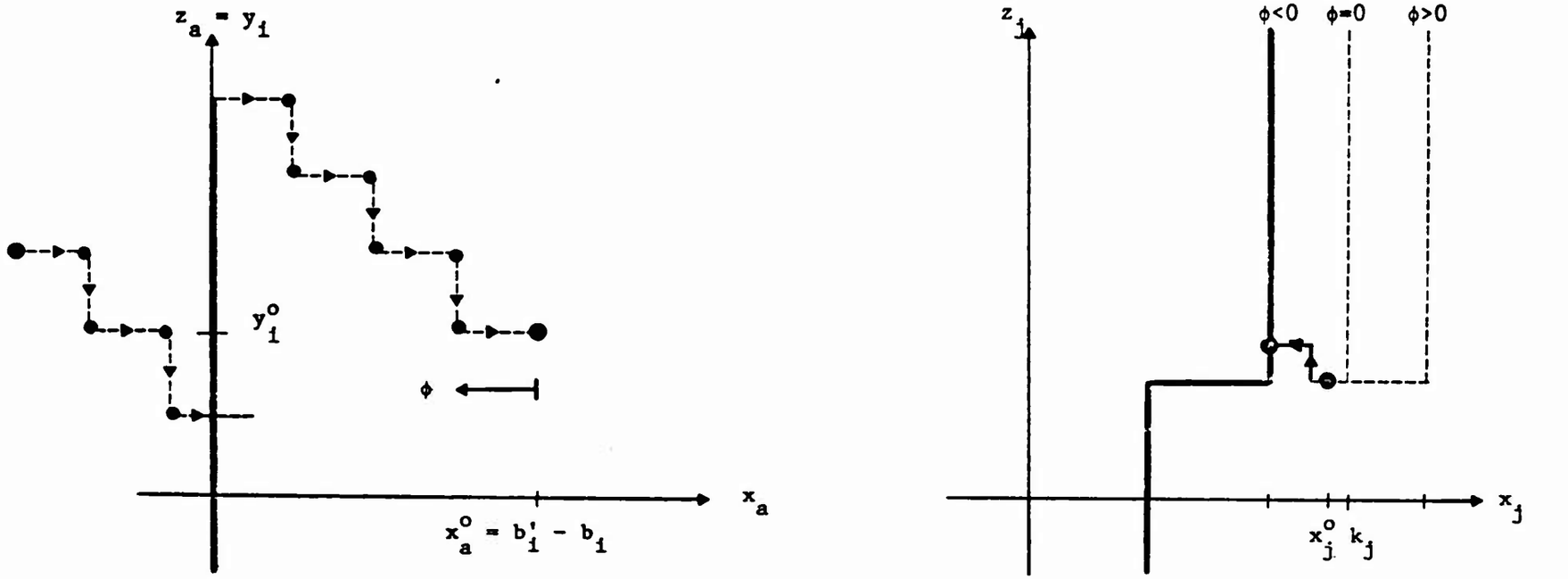
$$b_i' = b_i + \phi \tag{D17}$$

starting from an optimal solution $(x_j^o ; z_j^o)$. The corresponding equality constraint

can again be satisfied by adding an artificial variable x_a at level $x_a^o = b_i' - b_i$:

$$\sum_{j=1}^{n} a_{ij} x_j + x_a = b_i' \quad . \tag{D18}$$

This variable is now nonconforming with $z_a^o = y_i^o$, as shown in Figure D7a.



(a). Requirements Change

(b). Slack Inequality Change

FIGURE D7: OPTIMALITY DIAGRAMS IN RIGHT-HAND-SIDE PARAMETRIZATION

Since this is the only nonconforming variable, successive applications of the algorithm will always be minimizing ξ_a, and the dual variable y_i will increase as x_a decreases (ϕ increases). Horizontal displacements correspond to changes in $\mathcal{C}^*$, while vertical displacements cause changes in $\mathcal{D}^*$, while $(y_i; \phi)$ traces out the familiar breakpoint curve.

If the right-hand side change is in an inequality,

(D19) $$x_j \leq k_j + \phi$$

with $x_j^o > 0$, then no change will occur for ϕ positive, j remaining in B (Figure D7b). However, for some negative ϕ, j will become nonconforming in B^+, and a breakpoint curve similar to the requirements change above must be traced out.

Obvious remarks apply to vector changes in the $\{b_i\}$, changes in some ℓ_j, etc. (See also Section 9.)

I. Composite Methods

In the composite methods, such as the self-dual parametric algorithm [5], or the composite algorithm [13] one permits a starting solution with a basis B^o in which both

(D20) $x_j^o < 0$ for some (possibly all) $j \in B^o$ ("primal-infeasible")

(D21) $z_j^o > c_j$ for some (possibly all) $j \notin B^o$ ("dual-infeasible").

By a combination of the parametric methods of the last two paragraphs:

(a) Setting $x_{a_i}^o = x_j^o$, and $x_j = 0$ in equation i corresponding to the activity j in (D20), and then driving all x_{a_i} to zero;

and (b) Increasing c_j temporarily to the level z_j^o, and then decreasing all of them parametrically to zero, for all j in (D21).

one can, it is explained, always work with a primal-and-dual-feasible basic solution and proceed to the optimum "by successive application of either the primal or dual simplex rules." (Subsection A and C.) These parametric changes can all be worked on simultaneously, or one at a time (usually ignoring the others).

A much simpler explanation is given in terms of the optimality diagram. Because a basic solution is always given, the nonconforming activities are either in B^- or have ($z_j^o > c_j$ *and* $x_j^o = 0$) (Figure D8). Application of the algorithm takes B^o for $\mathcal{A}^o$, and always tries to increase ξ_j for all j in (D20) and (D21), either one at a time, or simultaneously. Wolfe [15] gives some suggestions for efficient choice of the ε_j, based on computational experience. A composite method is also described by Balinski and Gomory [2], which avoids degeneracy through a hierarchy of "sub-primal" and "sub-dual" tableaux.

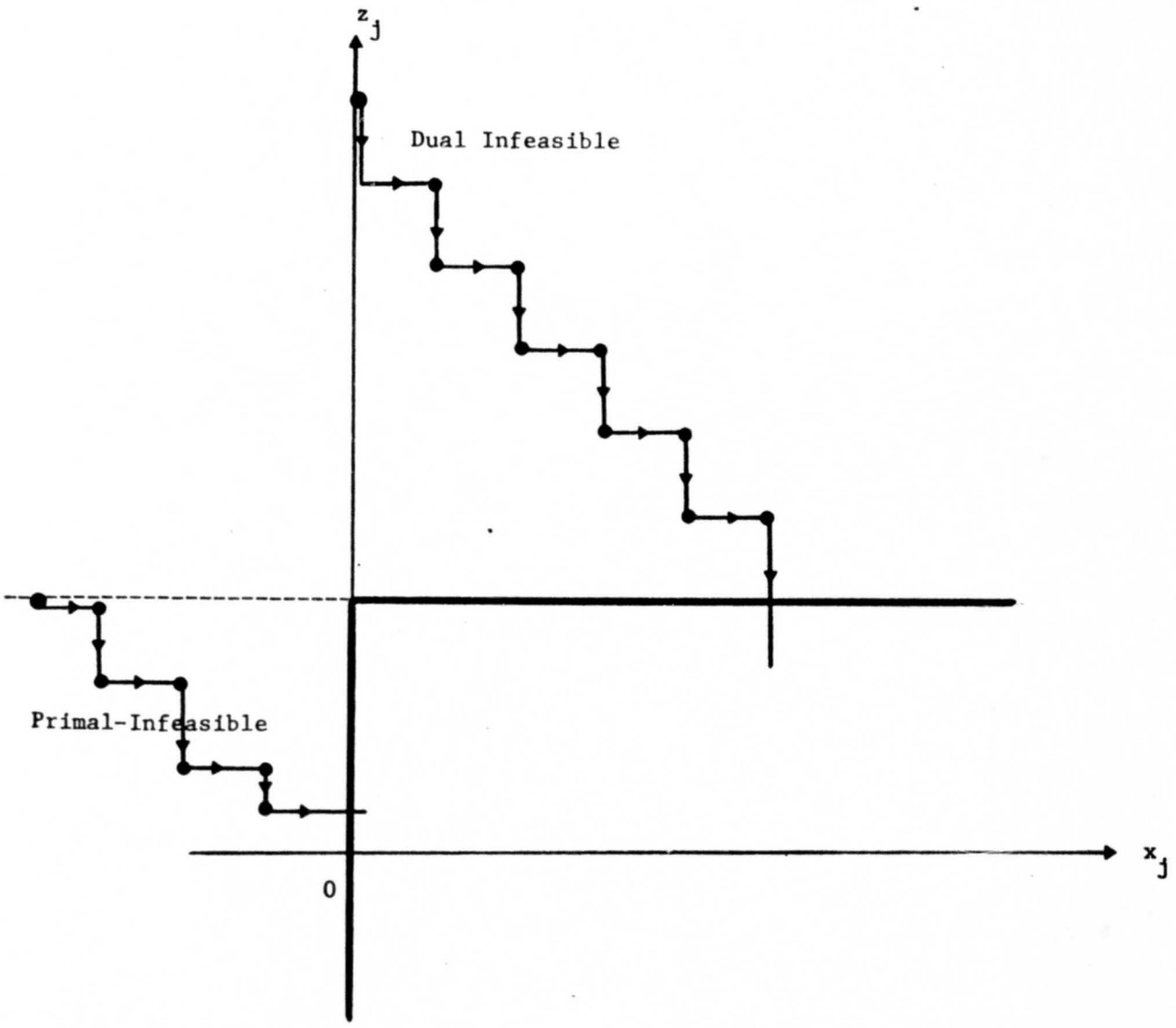


FIGURE D8: OPTIMALITY DIAGRAM IN THE COMPOSITE METHODS

APPENDIX E. AN EXAMPLE

As an example of the COMPLEX approach, consider the following program

$$\text{Min } \mathcal{C} = 2 \max (0, x_1 - 1) + 2 x_2 - x_3$$

$$x_1 + 2 x_2 + x_3 \geq 2$$

$$x_1 + 6 x_2 + 5 x_3 = -4$$

$$0 \leq x_1 \leq 3; -\infty \leq x_2 \leq +2; |x_3| \leq 4 .$$

We add variables x_4 and x_5 to the constraints, together with the bounds

$$-\infty \leq x_4 \leq 0; \; 0 \leq x_5 \leq 0 ,$$

and take the simple (but poor) starting solutions

$$x^0 = (0, 0, 0, 2, -4)$$
$$y^0 = (0, 0) \qquad \mathcal{C}^0 - \mathcal{B}^0 = 0 - 2M + 4 + 2M + 4M = 8M + 4$$
$$z^0 = (0, 0, 0, 0, 0)$$

Only activity #1 is initially conforming. The optimality diagrams are shown in Figure E1; note that no prior manipulation of the problem is needed to draw these diagrams.

The following steps represent a *possible* sequence of iterations following a COMPLEX procedure; the appropriate breakpoint trajectories are shown in Figure E1.

Iteration 1:

Max $\xi_5 \cdot \mathcal{J}^0 = \{4,5\}$. After two zero-level pivots, x_1 increases to its corner point, increasing x_5 , and decreasing x_6 . $\Delta^* = 2 \cdot \theta^* = \frac{1}{3}$ is limited by the downward movement $x_3 \cdot \mathcal{J}^1 = \{2,5\}$. Activity #3 is now conforming

$$\xi^* = (1, -\tfrac{1}{2}, 0, 0, 2); \quad \zeta^* = (2, 0, -2, 3, -1)$$

$$x^1 = (1, -\tfrac{1}{2}, 0, 0, 2); \quad z^1 = (1, 0, -1, 3/2, -\tfrac{1}{2})$$

$$y^1 = (3/2, -\tfrac{1}{2}); \quad C^1 - B^1 = 6M .$$

Iteration 2:

Max ξ_5 . A zero-level pivot leads to $\Delta^* = 0$ and the new basis $\mathcal{J}^2 = \{3, 5\}$. $\theta^* = \frac{1}{4}$ is limited by the upward movement of x_1 to a corner point.

$$\xi^* = (0, 0, 0, 0, 0, 0); \quad \zeta^* = (4, 4, 0, 5, -1)$$

$$x^2 = x^1 \quad ; \quad z^2 = (2, 1, -1, \tfrac{11}{4}, -3/4)$$

$$y^2 = (\tfrac{11}{4}, -3/4) \quad ; \quad C^2 - B^2 = 5M .$$

Iteration 3:

Max ξ_5 . Increasing x_1 drives activity #5 conforming, with $\Delta^* = 2$. Since all components of ζ^* are zero, θ^* may take on any value, but does not effect ξ^3 . $\mathcal{J}^3 = \{1,3\}$.

$$\xi^* = (\tfrac{1}{2}, 0, -\tfrac{1}{2}, 0, 2); \quad \zeta^* = (0, 0, 0, 0, 0)$$

$$x^3 = (3/2, -\tfrac{1}{2}, -\tfrac{1}{2}, 2, 0); \quad z^3 = z^2$$

$$y^3 = (\tfrac{11}{4}, -3/4); \quad C^3 - B^3 = 3M .$$

Iteration 4:

Min ξ_4 . Decreasing x_4 increases x_1 to its new corner point, also decreasing x_3. $\Delta^* = -6/5$, and $\mathcal{J}^4 = \{3,4\}$. $\theta^* = 5/4$ is limited only by the upward movement of x_2 into a conforming state

$$\xi^* = (3/2,\ 0,\ -\tfrac{3}{10},\ -\tfrac{6}{5},\ 0);\quad \zeta^* = \left(\tfrac{4}{5},\ \tfrac{4}{5},\ 0,\ 1,\ -\tfrac{1}{5}\right)$$

$$x^4 = (3,\ -\tfrac{1}{2},\ -\tfrac{4}{5},\ \tfrac{4}{5},\ 0) \qquad ;\ z^4 = (3,\ 2,\ -1,\ 4,\ -1)$$

$$y^4 = (4,\ -1);\quad \mathcal{C}^4 - \mathcal{B}^4 = \tfrac{4}{5}\,M\ .$$

Iteration 5:

Min ξ_4 . Increasing x_2 decreases both x_3 and x_4 , with $\Delta^* = -4/5$ as activity #4 becomes conforming. $\mathcal{A}^5 = \{2,\ 3\}$. There are no dual changes:

$$\xi^* = (0,\ 1,\ -6/5,\ -4/5,\ 0);\quad \zeta^* = (0,\ 0,\ 0,\ 0,\ 0)$$

$$x^5 = (3,\ \tfrac{1}{2},\ -2,\ 0,\ 0) \qquad ;\ z^5 = z^4 = (3,\ 2,\ -1,\ 4,\ -1)\ .$$

Since all states are now conforming, the above solutions are optimal, with $y^5 = y^4 = (4,\ -1)$ and $\mathcal{C}^5 - \mathcal{B}^5 = 0$.

Note, in particular, that the trajectories in conforming states need not be always in the same direction; indeed, they may sometimes pass through the same corner point several times.

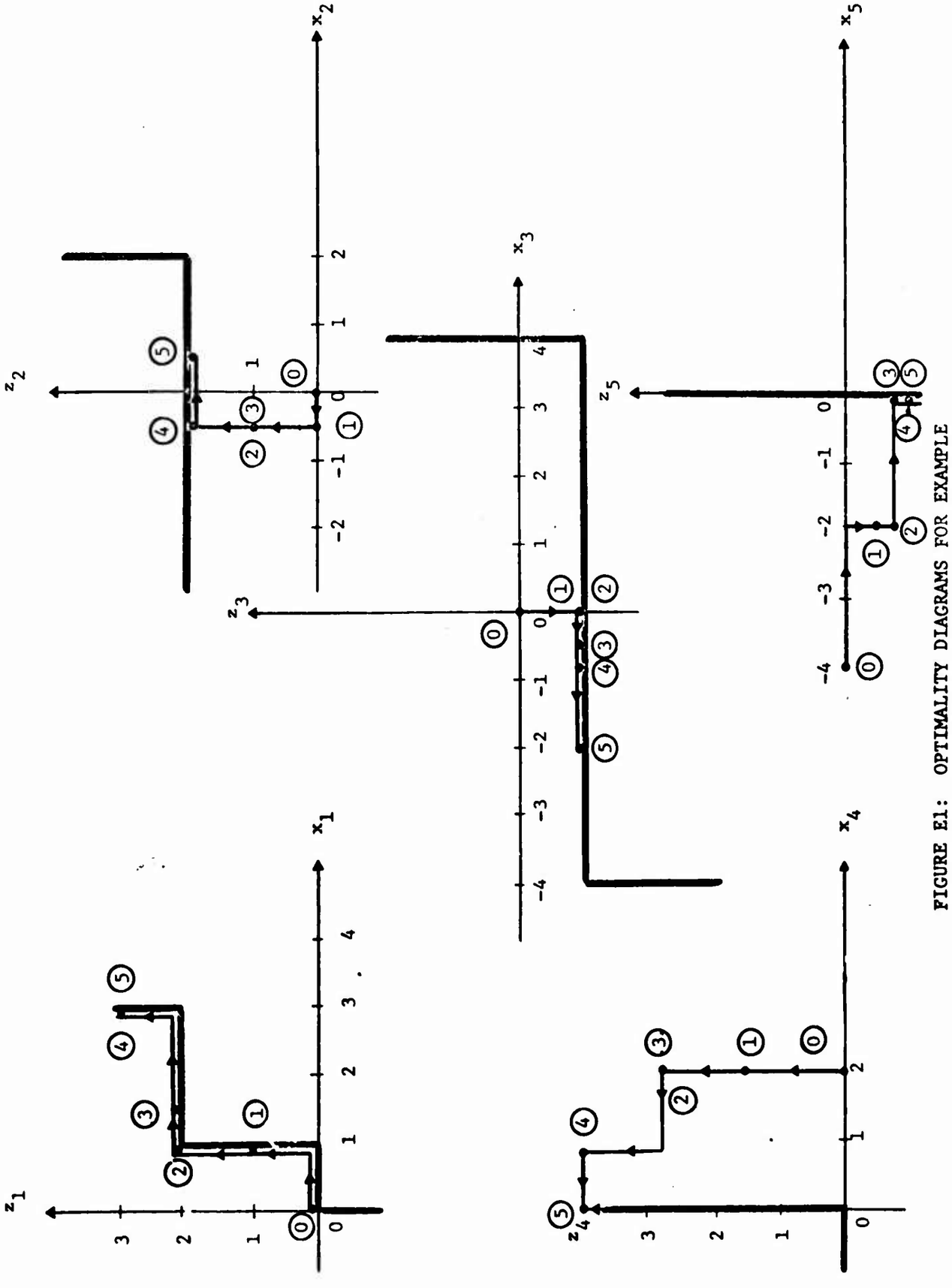


FIGURE E1: OPTIMALITY DIAGRAMS FOR EXAMPLE

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

University of California, Berkeley

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

A COMPLEMENTARY SLACKNESS, OUT-OF-KILTER ALGORITHM FOR LINEAR PROGRAMMING

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Research Report

5. AUTHOR(S) (Last name, first name, initial)

JEWELL, William S.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
March 1967	90	16

8a. CONTRACT OR GRANT NO.

Nonr-222(83)

b. PROJECT NO.

NR 047 033

c.

Research Project No.: RR 003 07 01

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

ORC 67-6

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. AVAILABILITY/LIMITATION NOTICES

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES Also supported by the Nat'l. Sci. Found. under Grant GP-7417, and the U.S. Army Res. Office-Durham, Contract DA-31-124-ARO-D-331.

12. SPONSORING MILITARY ACTIVITY

MATHEMATICAL SCIENCE DIVISION

13. ABSTRACT

SEE ABSTRACT

DD FORM 1 JAN 64 1473

Unclassified

Security Classification

Unclassified
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Linear Programming						
Out-of-kilter						
Complementary pivot						
Primal-dual						
Complementary slackness						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS), (S), (C), or (U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, roles, and weights is optional.

DD FORM 1 JAN 64 1473 (BACK)

Unclassified
Security Classification